Введение

В данной части книги читателю предлагается обзор изделий SIMATIC S7-300/400.

Программируемый контроллер SIMATIC S7-300/400 имеет модульную конструкцию. Модули, из которых составляется требуемая конфигурация контроллера, могут быть центральными (располагаться по соседству с CPU) или распределенными. В системах SIMATIC S7 распределенные входы/выходы (I/O) являются составной частью системы. CPU, имеющий различные области памяти, составляет основу оборудования системы для обработки программ пользователя. Загрузочная память (load memory) целиком содержит пользовательскую программу: части программы, выполняемые в любое заданное время (исполняемый модуль программы), находятся в рабочей памяти (work memory), обеспечивающей малое время доступа к данным, что предопределяет высокую скорость обработки программы.

STEP 7 – это программное обеспечение для программирования S7-300/400. Для организации работы по конфигурированию, программированию и тестированию программной части системы автоматического управления процессами служит утилита SIMATIC Manager. SIMATIC Manager – это приложение, работающее под управлением Windows 95/98/NT и содержащее все функции, необходимые для создания проекта. При необходимости SIMATIC Manager инициирует запуск других утилит, например, для конфигурирования станций, для инициализации модулей или для написания и тестирования программ.

Пользователь должен изложить свое программное решение для автоматизированной системы, используя языки программирования STEP 7. Программа SIMATIC S7 является структурированной программой, что означает, что она состоит из блоков, обладающих определенными функциями, соответствующими их положению в сетевой и иерархической структуре системы. Различные классы приоритетов позволяют располагать в определенном порядке прерывания исполняемой программы пользователя. STEP 7 работает с переменными различных типов, начиная с переменных двоичного типа (BOOL), с переменных численных форматов (INT или REAL) и заканчивая сложными типами, такими как массивы или структуры (комбинации переменных различных типов в форме единой переменной).

Первая глава книги содержит краткий обзор оборудования для программируемых контроллеров S7-300/400. Вторая глава книги содержит краткий обзор программного обеспечения STEP 7 для программирования. Описание строится на основе набора функций для STEP 7 версии 5.1.

Глава 3 "Программа SIMATIC S7" представляет собой введение в курс по наиболее важным элементам S7-программы и показывает способы программирования отдельных блоков программы на языках программирования STL и SCL. Функции и операторы языков STL и SCL описаны в последующих главах книги. Все описания сопровождаются пояснениями с использованием кратких примеров.

1 SIMATIC S7-300/400 программируемый контроллер
Структура программируемого контроллера;
распределенная периферия (I/O);
коммуникации;
адресация модулей;
области данных.

2 Программное обеспечение STEP 7 для программирования
SIMATIC Manager;
обработка проекта;
конфигурирование станций;
конфигурирование сети;
создание программ (таблица символов, редакторы программ);
включение интерактивного режима;
тестирование программы.

3 Программа SIMATIC S7
Обработка программы с классами приоритетов;
блоки программы;
адресация переменных;
программирование блоков с использованием STL и SCL;
переменные и константы;
типы данных (краткий обзор).

1 Программируемый контроллер SIMATIC S7-300/400

1.1 Структура программируемого контроллера

1.1.1 Компоненты

Программируемый контроллер SIMATIC S7-300/400 имеет модульную конструкцию и включает в себя следующие компоненты:

- Стойки (Rack):
  стойки используются для размещения в них модулей и для соединения последних друг с другом.
- Источник питания (PS – "power supply"):
  источник питания обеспечивает внутренние напряжения питания.
- Центральный процессор (CPU – "central processing unit"):
  центральный процессор используется для размещения и обработки программы пользователя.
- Интерфейсные модули (IM – "interface module"):
  интерфейсные модули используются для соединения стоек друг с другом.
- Сигнальные модули (SM – "signal module"):
  сигнальные модули используются для преобразования сигналов, поступающих от процесса, во внутренние сигналы для последующей обработки или в дискретные или аналоговые сигналы для управления приводами.
- Функциональные модули (FM – "function module"):
  функциональные модули не зависят от CPU, используются для выполнения сложных или зависящих от времени процессов.
- Коммуникационные процессоры (CP – "communication processor"):
  коммуникационные процессоры используются для связи с подсетями.
- Подсети:
  подсети используются для связи программируемых контроллеров друг с другом или с другими устройствами.

Программируемый контроллер (или станция) может состоять из нескольких стоек, которые связываются друг с другом посредством шины. Источник питания, CPU и I/O модули (модули SM, FM и CP) включаются в центральную стойку. Если для I/O модулей недостаточно места или необходимо часть или все I/O модули разместить вне центральной стойки, то в таких случаях используют дополнительные стойки – стойки расширения, которые соединяются с центральной стойкой посредством интерфейсных модулей (см. рис. 1). Также возможно подключение к станции распределенных входов/выходов (см. раздел 1.2, "Распределенные I/O").

Для связи модулей друг с другом в стойках служат две шины: шина входов/выходов (I/O или P-шина) и коммуникационная шина (или K-шина).

I/O-шина предназначена для высокоскоростного обмена входными и выходными сигналами, а коммуникационная шина обеспечивает обмен между модулями большими порциями данных. Коммуникационная шина соединяет CPU и интерфейс программатора (MPI) с функциональными модулями и коммуникационными процессорами.

1.1.2 Станция S7-300

Централизованная конфигурация

Программируемый контроллер S7-300 позволяет включить в центральную монтажную стойку до 8 входных/выходных модулей. Если такая однорядная конфигурация контроллера не является достаточной, то возможны два варианта расширения конфигурации при использовании CPU 314 или более мощных процессоров:

- или вариант двухрядной конфигурации, имеющей центральную стойку и одну стойку расширения (при использовании интерфейсных модулей IM 365 и с расстоянием до одного метра между стойками);
- или вариант конфигурации, состоящей максимально из 4 рядов, т.е. кроме центральной стойки, имеющей до 3 стоек расширения (при использовании интерфейсных модулей IM 360 и IM 361 и с расстоянием до десяти метров между стойками).

Вы можете задействовать максимум восемь модулей в стойке. Число модулей может быть ограничено также максимально допустимым током потребления на одну стойку, который составляет 1.2 A (для CPU 312 IFM максимально допустимый ток потребления составляет 0.8 A).

Модули связаны между собой внутренней шиной стойки, обеспечивающей функции P- и K-шин.

Локальный сегмент шины

Особую возможность при конфигурировании предоставляет использование прикладного модуля FM 356 из семейства компьютеров для автоматизации M7-300. Модуль FM-356 позволяет "разбить" интерфейсную шину модулей контроллера, чтобы получить контроль над оставшимися в "отсеченном сегменте шины" модулями для автономного управления ими. В данном случае ограничивающим фактором также являются такие параметры, как число модулей и суммарная потребляемая ими мощность.

Внешние условия для изделий SIMATIC

Модули SIMATIC S7-300 допускают использование в жестких внешних условиях. Они имеют расширенный температурный рабочий диапазон: (-25…+60)°C, повышенную вибрационную и ударную стойкость, соответствующие стандарту IEC 68 часть 2-6; удовлетворяют требованиям по влагостойкости, устойчивости к образованию конденсата и инея согласно IEC 721-3-3 Class 3 K5, также как и требованиям стандарта для ж/д транспорта по EN 50155 (в скором будущем). Остальные характеристики стандартны.

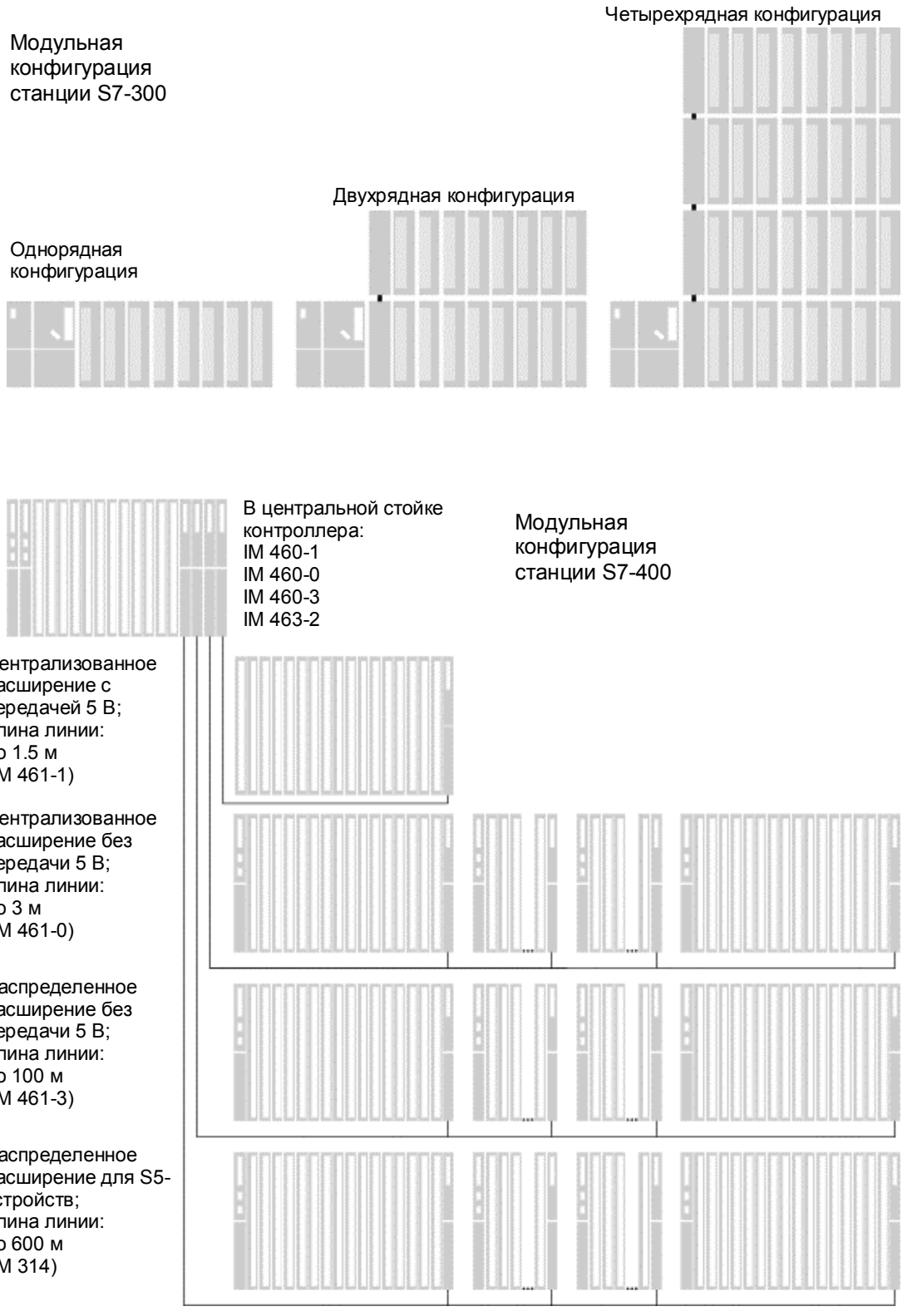


Рис. 1.1 Конфигурация аппаратной части для S7-300/400

1.1.3 Станция S7-400

Централизованная конфигурация

Программируемый контроллер S7-400 имеет следующий состав: центральная монтажная стойка на 18 или 9 слотов (соответственно UR1 или UR2), модуль блока питания и модуль CPU, которые могут сами занимать от одного до нескольких слотов (посадочных мест в монтажной стойке). Интерфейсные модули IM 460-1 и IM 461-1 позволяют использовать одну стойку расширения с подачей в нее от центральной стойки 5-вольтового питающего напряжения с длиной линии до 1.5 метров между стойками. Кроме того, с помощью интерфейсных модулей IM 460-0 и IM 461-0 могут быть использованы от одной до 4 стоек расширения с длиной шины связи с центральной монтажной стойкой до 3 метров. И наконец, с помощью интерфейсных модулей IM 460-3 и IM 461-3 могут быть использованы от одной до 4 стоек расширения с длиной шины связи с центральной монтажной стойкой до 100 метров.

Максимальное количество подсоединяемых к центральной монтажной стойке стоек расширения составляет 21 единицу. Для распознавания стоек расширения необходимо их количество задавать с помощью кодирующего переключателя на приемном интерфейсном модуле.

Внутренняя шина состоит из параллельной P- и последовательной K-шин. Стойки расширения ER1 и ER2 (соответственно на 18 и 9 слотов) предназначены для "простых" сигнальных модулей, которые не могут генерировать аппаратные прерывания, не имеют 24-вольтового питающего напряжения по P-шине, не имеют резервного питания и не имеют связи по K-шине. K-шина используется в стойках UR1, UR2 и CR2 или в случае применения их в качестве центральных монтажных стоек, или в случае применения их в качестве монтажных стоек расширения с номерами от 1 до 6.

Присоединение сегментированной стойки

Особую возможность при конфигурировании предоставляет использование сегментированной монтажной стойки CR2. Сегментированная монтажная стойка CR2 позволяет использовать два центральных процессора с общим для них модулем питания. При этом оба CPU могут сохранять свою функциональную обособленность, так как имеют отдельные P-шины со своими собственными сигнальными модулями, и могут в то же время обмениваться данными посредством K-шины.

Мультипроцессорный режим

В программируемом контроллере S7-400 возможно организовать мультипроцессорный режим с участием нескольких (до четырех единиц) специальных CPU. При этом в такой станции каждый модуль назначается только одному из CPU (соответственно с его адресацией и прерываниями). За более детальной информацией обратитесь к разделам 20.3.6 "Мультипроцессорный режим" и 21.6 "Прерывание мультипроцессорного режима".

Модули SIMATIC S5

Интерфейсный модуль IM 463-2 позволяет подключать к программируемому контроллеру S7-400 устройства расширения SIMATIC S5 (EG 183U, EG 185U, EG 186U, ER 701-2 и ER 701-3), а также позволяет централизованное расширение устройств расширения. Интерфейсный модуль IM 314 в устройствах расширения SIMATIC S5 используется для обеспечения функций связи. Вы можете использовать все аналоговые и дискретные модули, допустимые в этих устройствах расширения. В контроллере S7-400 могут использоваться от одного до четырех интерфейсных модулей IM 463-2; а к каждому из интерфейсных модулей IM 463-2, таким образом, могут быть подключены от одного до четырех устройств расширения S5 в распределенной конфигурации.

Резервирование на основе программного обеспечения

Используя стандартные компоненты SIMATIC S7-300/400, Вы можете создать резервированную на основе программного обеспечения систему с ведущей станцией управления процессом и резервной станцией, которая принимает на себя управление процессом в случае выхода из строя ведущей станции.

Устойчивость к сбоям системы управления с резервированием на основе программного обеспечения подходит для так называемых "медленных процессов", так как переключение управления на резервную станцию может потребовать нескольких секунд, в зависимости от конфигурации программируемых контроллеров. Сигналы, поступающие от процесса, "замораживаются" на время перехода управления к резервной станции. Резервная станция будет продолжать работу по управлению процессом, используя последние корректные данные, полученные ведущей станцией.

Резервирование входных/выходных модулей обеспечивается с помощью распределенной периферии (I/O) (ET 200M с интерфейсным модулем IM 153-3 для резервирования PROFIBUS-DP). В системе управления может быть сконфигурировано заказное (опционное) программное обеспечение для резервирования ("Software Redundancy").

Отказоустойчивый контроллер SIMATIC S7-400H

SIMATIC S7-400H – это отказоустойчивый программируемый контроллер с резервированной конфигурацией, имеющей две центральные стойки, каждая с H CPU и с модулем синхронизации для сравнения данных с волоконно-оптическим кабелем. Оба регулятора работают в режиме "горячего резервирования"; в случае отказа неповрежденный контроллер в одиночку продолжает выполнять функции управления после плавного автоматического переключения на резервный режим работы.

Входы/выходы могут обеспечивать обычный нормальный доступ (одноканальная, односторонняя конфигурация) или расширенный доступ (одноканальная переключаемая конфигурация с ET 200M). Связь осуществляется по простой или по резервной шине.

Программа пользователя точно такая же, как и для не резервированного контроллера; функции резервирования выполняются исключительно аппаратно и не заметны для пользователя. Для конфигурирования системы требуется заказное (опционное) программное обеспечение "S7-400H".

1.1.4 Области памяти CPU

Память пользователя

На рисунке 1.2 показаны области памяти CPU, имеющие значение для Вашей программы. Программа пользователя собственно располагается в двух областях, а именно в загрузочной памяти (load memory) и в рабочей памяти (work memory).

Загрузочная память (load memory) конструктивно может быть частью CPU или может быть в виде встраиваемого отдельного модуля памяти. Вводимая пользователем программа, включая данные конфигурирования, располагается в загрузочной памяти (load memory) и в оперативной памяти.

Рабочая память (work memory) конструктивно является частью CPU и представляет собой быструю RAM-память. В оперативной памяти содержатся релевантные части программы пользователя: собственно код программы и данные пользователя. Здесь "релевантность" означает, что в эту память загружается код, описывающий существующие объекты, но это не предполагает обязательность вызова отдельных блоков этого кода для обработки.

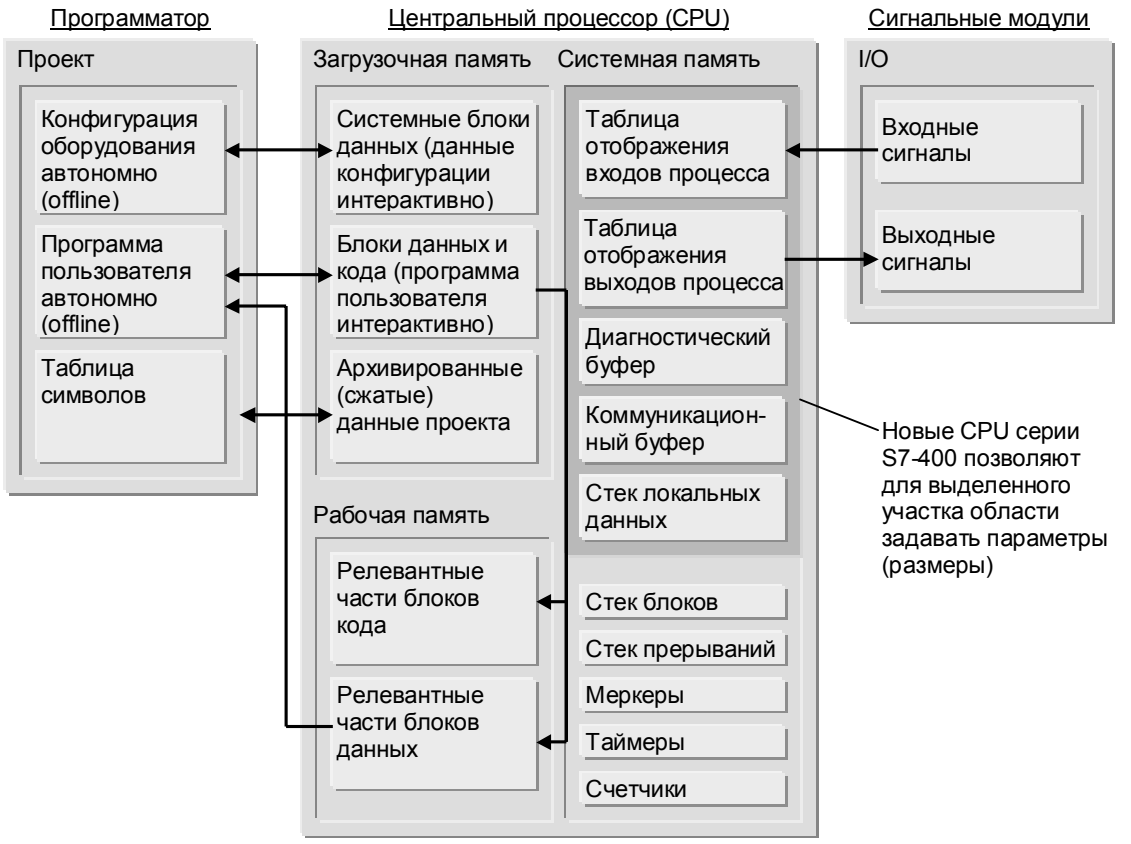


Рис. 1.2 Области памяти CPU

Из программатора программа пользователя целиком, включая данные конфигурации, пересылается в загрузочную память (load memory). Операционная система CPU копирует "релевантные" (см. выше) части программного кода и данных в рабочую память (work memory). Когда программа считывается программатором из CPU, блоки выбираются из загрузочной памяти (load memory) с текущими значениями адресов данных из рабочей памяти (work memory) (для получения более подробной информации см. разделы 2.6.4 "Загрузка программы пользователя в CPU" и 2.6.5 "Обработка блоков").

Если загрузочная память (load memory) построена на основе RAM-памяти, то необходимо использовать дополнительную батарею для резервирования питания, чтобы обеспечивать сохранность программы пользователя в случае отказа штатной системы питания CPU. Если загрузочная память (load memory) выполнена на основе встроенной EEPROM-памяти или внешнего модуля EPROM флэш-памяти, то CPU может использоваться без дополнительного резервирования питания батареей.

Загрузочная память (load memory) в CPU 3xxIFM состоит из RAM и EEPROM компонентов. Вы можете загрузить Вашу программу в RAM-область для ее тестирования, а затем протестированную программу можете посредством команд меню сохранить во внутренней EEPROM-памяти, где она не будет зависеть от отказов блока питания.

Загрузочная память (load memory) в CPU для S7-300 (за исключением CPU 318) состоит из встроенной RAM-памяти, которая может целиком вмещать программу. При этом Вы можете использовать модуль EPROM флэш-памяти в качестве носителя для данных и программ пользователя или в качестве памяти, защищенной от сбоев питания.

В CPU для S7-300 текущие значения из областей памяти пользователя (блоки данных) и системной памяти (меркеры, таймеры, счетчики) могут содержаться в энергонезависимой форме. Таким образом пользователь может сохранять свои данные без применения резервной батареи в условиях возможных перебоев электропитания.

Загрузочная встроенная RAM-память в CPU для S7-400 предназначена для маленьких программ или для модифицирования отдельных блоков. Если полная программа по объему больше, чем встроенная загрузочная память (load memory), то Вам потребуется модуль RAM-памяти для тестирования программы. Вы можете использовать модуль EPROM флэш-памяти в качестве носителя для данных или программы пользователя с бесперебойным питанием.

В новых CPU для S7-400 рабочая память (work memory) может наращиваться с помощью установки дополнительных модулей.

Начиная с STEP 7 V5.1, используя соответствующие CPU для S7-400, Вы можете сохранять все данные проекта в виде архивированного сжатого файла в загрузочной памяти (load memory) CPU (см. раздел 2.2.2 "Управление, перекомпоновка и архивирование").

1.1.5 Модуль памяти

Существуют два типа модулей памяти: модули RAM-памяти и модули EPROM флэш-памяти.

Если необходимо просто увеличить загрузочную память (load memory), то используйте для этого модуль RAM-памяти (например, в CPU для S7-400). Модуль RAM-памяти позволит Вам целиком обновлять программу пользователя в интерактивном режиме. При этом необходимо помнить, что модули RAM-памяти теряют всю записанную в них информацию при вынимании их из слота.

Если Вам необходимо защитить от стирания из-за возможного сбоя в цепях питания пользовательскую программу, включая данные конфигурации и параметры модулей, то используйте модуль EPROM флэш-памяти. При использовании такого модуля памяти загружайте целиком в него программу в автономном режиме, установив модуль EPROM флэш-памяти в программатор. При использовании соответствующего CPU Вы в интерактивном режиме сможете загружать в модуль свою программу пользователя, установив модуль EPROM флэш-памяти непосредственно в такой CPU.

1.1.6 Системная память

Системная память содержит адреса (переменные), к которым Вы обращаетесь в своей программе. Все адреса объединяются в области (адресное пространство), содержащие определенное, зависящее от конкретного CPU, число адресов. Адреса эти могут, например, принадлежать входам, используемым для опроса состояния сигналов от кнопок или конечных переключателей, или выходам, используемым для управления контакторами (реле) или лампами.

Системная память CPU содержит следующие адресные области:

- Входы (I):

  входы формируют "отображение процесса по входам" дискретных входных модулей.

- Выходы (Q):

  выходы формируют "отображение процесса по выходам" дискретных выходных модулей.

- Меркеры (M):

  меркеры хранят информацию, доступную из любой точки программы.

- Таймеры (T):

  таймеры хранят информацию, определяющую параметры времени для функций ожидания и мониторинга.

- Счетчики (C):

  счетчики хранят информацию для функции прямого и обратного счета.

- Временные локальные данные (L)

  временные локальные данные используются в качестве динамических промежуточных буферов при обработке блоков. Временные локальные данные располагаются в L-стеке, который динамически занимается и высвобождается CPU при выполнении программы.

Буквы, заключенные в скобки в выше приведенных пунктах перечисления адресных областей, представляют собой аббревиатуры, используемую в мнемониках программы при различных способах задания адресов. Вы можете также назначать символ для каждой переменной и затем использовать этот символ вместо идентификатора адреса.

В системной памяти также содержатся буферы для коммуникационных заданий и системных сообщений (буфер диагностики). Размеры этих буферов данных, также как и размеры областей хранения отображения процесса по входу и выходу, в новых центральных процессорах для S7-400 может определять пользователь.

1.2 Распределенные I/O (входы/выходы)

Сеть PROFIBUS-DP обеспечивает стандартный интерфейс для передачи преимущественно двоичных данных процесса между "интерфейсным модулем" в центральном программируемом контроллере и приборами полевого уровня. Этот "интерфейсный модуль" называется "ведущим DP-устройством" (DP-master), а приборы полевого уровня называются "ведомыми DP-устройствами" (DP-slave). Распределенные входы/выходы (I/O) относятся к модулям, подключенным посредством PROFIBUS-DP к ведущему модулю PROFIBUS. PROFIBUS-DP совместим со стандартом EN 50170 и является независимым от производителей стандартом для подключения стандартных ведомых DP-устройств.

За дополнительной информацией обратитесь к разделу 1.3.2 "Подсети".

Ведущее DP-устройство и все управляемые им ведомые DP-устройства образуют "систему ведущего DP-устройства" (DP-master system). В одном сегменте сети может быть до 32 станций и максимально до 127 станций может быть в сети всего. Ведущему DP-устройству соответствует определенное число, управляемых им ведомых DP-устройств. Пользователь может также подключать к сети PROFIBUS-DP программатор, также как и, например, устройства для обеспечения человеко-машинного интерфейса, устройства ET 200 или ведомые DP-устройства SIMATIC S5.

1.2.1 Система ведущего DP-устройства

Система с одним ведущим DP-устройством (mono master system)

Сеть PROFIBUS-DP обычно используется как система с одним ведущим DP-устройством ("mono master system"); в такой сети одно ведущее DP-устройство управляет несколькими ведомыми DP-устройствами. В этом режиме ведущее DP-устройство является единственным ведущим устройством на шине, за исключением случаев, когда возможно временное подключение программатора в качестве устройства диагностики и обслуживания. Ведущее DP-устройство и все управляемые им ведомые DP-устройства образуют "систему ведущего DP-устройства" ("DP-master system") (см. ниже рис. 1.3).

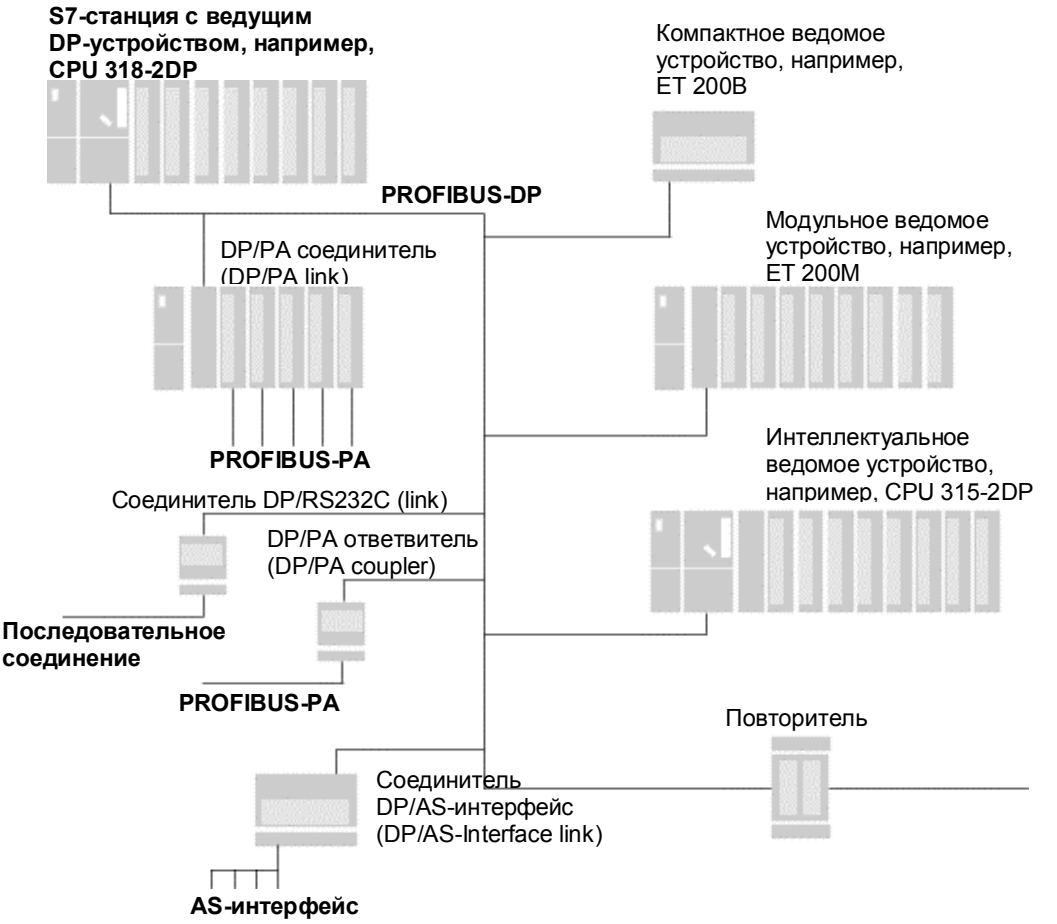


Рис. 1.3 Компоненты системы ведущего DP-устройства (DP-master system)

Система с несколькими ведущими DP-устройствами (multi master system)

Вы можете также установить в одной сети PROFIBUS-DP несколько ведущих DP-устройств ("multi master system"). Тем не менее, это приводит к уменьшению времени отклика в такой сети для каждого сегмента с ведущим DP-устройством; это важно, так как при этом режиме, когда одно ведущее DP-устройство инициализирует "свои" ведомые DP-устройства, другое ведущее DP-устройство не имеет доступа к "своим" ведомым DP-устройствам при попытке их инициализировать и т. д.

Несколько систем ведущих DP-устройств в каждой станции

Вы можете уменьшить время отклика, если в системе ведущего DP-устройства будет уменьшено число ведомых DP-устройств. Так как в одной S7-станции возможно установить несколько ведущих DP-устройств, Вы можете распределить ведомые DP-устройства всей станции между их системами. В мультипроцессорном режиме каждый CPU имеет свою собственную "систему ведущего DP-устройства".

1.2.2 Ведущее DP-устройство (DP Master)

Ведущее DP-устройство (DP Master) является активным узлом сети PROFIBUS. Это ведущее устройство циклически обменивается данными со "своими" ведомыми DP-устройствами. Ведущим DP-устройством может быть:

- CPU с встроенным интерфейсом ведущего DP-устройства или с вставленным интерфейсным модулем (например, CPU 315-2DP, CPU 417).
- Интерфейсный модуль в соединении с CPU (например, IM 467).
- Коммуникационный процессор CP в соединении с CPU (например, CP 342-5, CP 443-5).

Существуют "ведущие DP-устройства 1 класса", предназначенные для обмена данными при обработке процесса, и "ведущие DP-устройства 2 класса", предназначенные для обслуживания и диагностики (например, программаторы).

1.2.3 Ведомые DP-устройства (DP Slaves)

Ведомые DP-устройства (DP Slaves) являются пассивными узлами сети PROFIBUS. В SIMATIC S7 различают следующие ведомые DP-устройства:

- Компактные ведомые устройства, которые ведут себя как отдельные модули по отношению к ведущему DP-устройству.
- Модульные ведомые устройства, состоящие из нескольких (под)модулей.
- Интеллектуальные ведомые устройства, содержащие программу управления для собственных подчиненных модулей.

Компактные ведомые PROFIBUS DP-устройства

Примерами компактных ведомых DP-устройств могут послужить следующие устройства:

ET 200B (в версии для дискретных входных/выходных модулей или аналоговых входных/выходных модулей, имеющий степень защиты IP 20 и максимальную скорость передачи данных, равную 12 Мбит/с);

ET 200C (имеющий конструкцию для условий эксплуатации, отвечающих стандарту IP 66/67, имеющий варианты исполнения для дискретных входных/выходных модулей и аналоговых входных/выходных модулей, имеющий максимальную скорость передачи данных, равную 1,5 Мбит/с или 12 Мбит/с);

ET 200L-SC (дискретно-модульной конструкции с возможностью свободного комбинирования дискретных входных/выходных модулей и аналоговых входных/выходных модулей, имеющий степень защиты IP 20 и максимальную скорость передачи данных, равную 1,5 Мбит/с);

Шинные шлюзы, такие как соединитель DP/AS-I (DP/AS-I Link), ведут себя как компактные ведомые DP-устройства в сети PROFIBUS-DP.

Модульные ведомые PROFIBUS DP-устройства

Примером модульных ведомых DP-устройств может служить устройство ET 200M. Его конструкция аналогична конструкции станции S7-300, имеет профильную шину DIN, модуль блока питания, интерфейсный модуль IM 153 на месте CPU и до 8 сигнальных модулей (SM) или функциональных модулей (FM). Скорость передачи данных составляет от 9,6 кбит/с до 12 Мбит/с).

ET 200M может также иметь в своем составе *активные шинные модули*, если ведущим DP-устройством является станция S7-400. Это означает, что входные/выходные модули S7-300 могут быть добавлены в стойку или удалены из нее в то время, когда она работает с включенным питанием. При этом остающиеся в стойке модули продолжают работать. И при этом при установке модулей в такой стойке не накладывается больше требование плотного их расположения, т.е. не запрещается пропускать слоты при установке модулей в монтажную стойку.

ET 200M может быть также использован с интерфейсным модулем IM 153-3 как ведомое DP-устройство в *резервной шине.* Интерфейсный модуль IM 153-3 имеет два соединителя: один - для подключения к ведущему DP-устройству в ведущей (основной) станции и второй - для подключения к ведущему DP-устройству в резервной станции.

Интеллектуальные ведомые PROFIBUS DP-устройства

Примером интеллектуальных (программируемых) ведомых DP-устройств может послужить станция S7-300, в которой задействован CPU с DP-интерфейсом, который может быть переключен в режим ведомого (slave) устройства (как например, CPU 315-2DP), а также станция S7-300 с коммуникационным процессором CP 342-5 в режиме ведомого (slave) устройства.

Как интеллектуальное ведомое DP-устройство может работать также ET 200X с базовым модулем BM 147/CPU. Он включает в себя базовый модуль и до 7 модулей расширения. В качестве базовых модулей Вы можете использовать "пассивные" базовые модули с дискретными входами или выходами или же Вы можете использовать "интеллектуальный" базовый модуль BM 147/CPU, способный обрабатывать S7-программу пользователя. Модули расширения могут быть представлены дискретными входными/выходными модулями и аналоговыми входными/выходными модулями, а также модулями нагрузок (load feeders), служащими для подключения и защиты любых трехфазных нагрузок для переменного тока напряжением до 400 В мощностью до 5,5 кВт.

Базовые модули обеспечивают передачу данных со скоростями от 9,6 кбит/с до 12 Мбит/с.

1.2.4 Подключение к PROFIBUS-PA

PROFIBUS-PA

PROFIBUS-PA ("Process Automation" [автоматизация процесса]) – это шинная система для организации процессов как в взрывоопасных зонах (или в так называемых Ex-зонах, например, в таких областях, как химическая промышленность), так и в взрывобезопасных зонах (например, в пищевой отрасли).

Протокол для PROFIBUS-PA опирается на стандарт EN 50170, том 2 (PROFIBUS-DPA); способ передачи данных отвечает стандарту IEC 1158-2.

Существуют два возможных способа соединения PROFIBUS-DP и PROFIBUS-PA:

- DP/PA ответвитель (DP/PA coupler), применяемый в случае, когда сеть PROFIBUS-DP может работать со скоростью передачи данных, равной 45,45 кбит/с.
- DP/PA соединитель(DP/PA link), обеспечивающий согласование разных скоростей обмена в PROFIBUS-DP и PROFIBUS-PA.

DP/PA ответвитель (DP/PA coupler)

DP/PA ответвитель (DP/PA coupler) позволяет подключать PA-приборы полевого уровня к сети PROFIBUS-DP. В сети PROFIBUS-DP DP/PA ответвитель имеет статус ведомого DP-устройства со скоростью обмена данными, равной 45,45 кбит/с. К одному DP/PA ответвителю могут быть подключены до 31 PA-прибора полевого уровня. Такая совокупность "полевых" приборов образует сегмент PROFIBUS-PA со скоростью обмена данными, равной 31,25 кбит/с. Взятые все вместе сегменты PROFIBUS-PA образуют шинную систему PROFIBUS-PA общего использования (shared).

DP/PA ответвитель может быть в двух вариантах исполнения:

- DP/PA ответвитель не-Ex версии с выходным током до 400 мА и
- DP/PA ответвитель Ex версии с выходным током до 100 мА.

DP/PA соединитель (DP/PA link)

DP/PA соединитель (DP/PA link) позволяет подключать PA-приборы полевого уровня к сети PROFIBUS-DP и обеспечивать скорость обмена данными от 9,6 кбит/с до 12 Мбит/с. DP/PA соединитель имеет в своем составе интерфейсный модуль IM 157 и до 5 единиц DP/PA-ответвителей, связанных между собой посредством шинных соединителей (коннекторов) SIMATIC S7. Шинная система, состоящая из сегментов PROFIBUS-PA, образует ведомое PROFIBUS-PD устройство. К одному DP/PA соединителю Вы можете подключить до 31 PA-прибора полевого уровня.

SIMATIC DPM

SIMATIC DPM (Process Device Manager [менеджер настройки приборов автоматики], ранее: "SIPROM") – это независимая от поставщика утилита, служащая для задания параметров, отладки, запуска и диагностики интеллектуальных приборов полевого уровня с возможностью работы с протоколами PROFIBUS-PA или HART (Highway Addressable Remote Transducers [протокол обмена с удаленными адресуемыми датчиками-преобразователями]). Для задания параметров датчикам-преобразователям используется программное средство DDL (Device Description Language [язык параметризации приборов]).

Вы можете работать с SIMATIC DPM как в "автономном" режиме, предназначенном для работы под Windows 9x/NT, или как с составной частью системы STEP 7.

1.2.5 Подключение к AS-интерфейсу

AS-интерфейс

AS-интерфейс ("Actuator-Sensor Interface" ("AS-i") [интерфейс привод-датчик]) – это сетевая система для обмена данными с оборудованием процесса нижнего уровня в системе управления. Ведущее устройство AS-i может управлять группой, включающей до 31 единиц ведомых устройств AS-i. Управление обеспечивается по двухпроводной AS-i-линии, по которой передаются как питающее напряжение, так и информационные сигналы. Ведомыми устройствами AS-i могут быть приводы или датчики с шинной организацией или AS-i модули, к которым подключено до 8 двоичных ("normal" - "нормальных") датчиков или приводов.

Сегмент AS-i может иметь длину до 100 м, однако длина сегмента может быть увеличена вдвое при применении повторителя (при этом ведомые устройства AS-i и источники питания должны присутствовать на обоих концах) или при применении расширителя (при этом ведомые устройства AS-i и источник питания должны быть только на линии, идущей от ведущего устройства AS-i).

Ведущее устройство AS-i (AS-i master)

Ведущее устройство AS-i (AS-i master) обновляет свои данные и данные всех подключенных к нему ведомых устройств AS-i на интервале времени, не превышающем 5 мс. Вы можете подключать AS-i шину непосредственно к SIMATIC S7 с помощью коммуникационного процессора CP 342-2 или к сети PROFIBUS-DP с помощью использования DP/AS-интерфейсного соединителя (см. рис. 1.4).

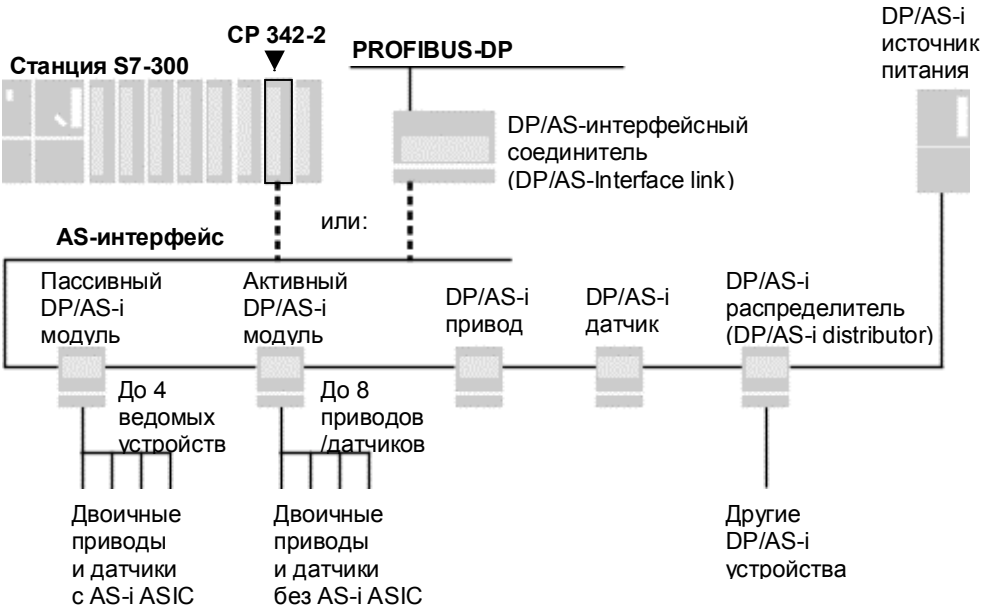


Рис. 1.4 Подключение к SIMATIC S7 шинной системы AS-i

Коммуникационный процессор **CP 342-2** в качестве ведущего устройства AS-i может быть использован в станции S7-300 или в станции ET 200M. Он поддерживает два рабочих режима:

В *стандартном режиме* CP 342-2 ведет себя как входной/выходной модуль. Он занимает 16 входных байтов и 16 выходных байтов в аналоговом адресном пространстве (начиная с 256). Ведомые устройства AS-i параметризуются данными в CP, заданными по умолчанию.

В *расширенном режиме* реализуется полный набор функциональных возможностей ведущего устройства AS-i. Если используется блок FC, то вызовы ведущего устройства могут выполняться из программы пользователя в добавление к возможностям стандартного режима (передача параметров во время работы, проверка заданной/фактической конфигурации, тестирование и диагностика).

DP/AS-интерфейсный соединитель (DP/AS-Interface link) обеспечивает подключение AS-i приводов и AS-i датчиков к сети PROFIBUS-DP. В сети PROFIBUS-DP соединитель имеет статус модульного ведомого DP-устройства, а в сети AS-интерфейс он имеет статус ведущего AS-i устройства, которое может контролировать до 31 ведомого AS-i устройства. При максимальном числе ведомых AS-i устройств DP/AS-интерфейсный соединитель занимает 16 входных байтов и 16 выходных байтов. Скорость передачи - до 12 Мбит/с.

DP/AS-интерфейсный соединитель может выполняться в двух вариантах: для жестких условий эксплуатации (версия 65) со степенью защиты IP 66/67 и (версия 20) со степенью защиты IP 20 с возможностью установки дополнительного командного интерфейса, при котором и для входов и для выходов диапазон адресов возрастает до 20 байт.

1.2.6 Подключение к последовательному интерфейсу

Соединитель **PROFIBUS-DP/RS 232C** (**PROFIBUS-DP/RS 232C link)** является конвертором для соединения интерфейса RS 232C (V.24) и PROFIBUS-DP. Посредством соединителя DP/RS 232C устройства с интерфейсом RS 232C могут быть подключены к PROFIBUS-DP. Соединитель DP/RS 232C поддерживает протоколы 3964R и ASCII.

Соединитель DP/RS 232C обеспечивает подключение приборов способом "точка к точке". Данные передаются с сохранением консистентности в обоих направлениях. В фрейме передается до 224 байт данных пользователя.

Скорость передачи данных по PROFIBUS-DP достигает 12 Мбит/сек; RS 232C обеспечивает скорость передачи данных до 38,4 кбит/с без контроля по четности, с проверкой на четность или нечетность, 8 битов данных плюс 1 стоп-бит.

1.3 Коммуникации (Communications)

Коммуникации обеспечивают обмен данными между программируемыми модулями - это встроенный компонент SIMATIC S7. Почти все коммуникационные функции управляются операционной системой. Обмен данными может быть организован без какого-либо дополнительного оборудования посредством только одного соединительного кабеля между двумя CPU. При использовании модулей CP можно создавать мощные сети и с легкостью подключать к ним системы сторонних (кроме SIEMENS) производителей оборудования.

SIMATIC NET - более широкое понятие, включающее в себя понятие коммуникаций SIMATIC. SIMATIC NET представляет собой информационный обмен между программируемыми контроллерами, а также между программируемыми контроллерами и устройствами HMI (человеко-машинный интерфейс). С SIMATIC могут быть реализованы различные варианты функций связи в зависимости от поставленной задачи.

1.3.1 Введение

На рисунке 1.5 показаны наиболее важные объекты связи. Может возникнуть задача реализации обмена данными между станциями SIMATIC или аппаратами сторонних (отличных от SIEMENS) производителей. В этом случае необходимы модули с функцией связи. С помощью SIMATIC S7 все CPU обеспечиваются MPI интерфейсом, с помощью которого они могут связываться друг с другом. Кроме того, для связи могут быть применены коммуникационные процессоры (CP), выполняющие обмен данными с высокой пропускной способностью и с различными протоколами обмена.

Модули могут быть связаны сетью. Сеть - это аппаратное соединение между узлами связи (коммуникационными узлами).

Обмен данными происходит посредством "соединения" в соответствии со специальным планом обработки данных ("служба обмена"), который основывается на специальной процедуре ("протокол"). Например, S7-соединение является стандартом для S7-модулей с функциями связи.

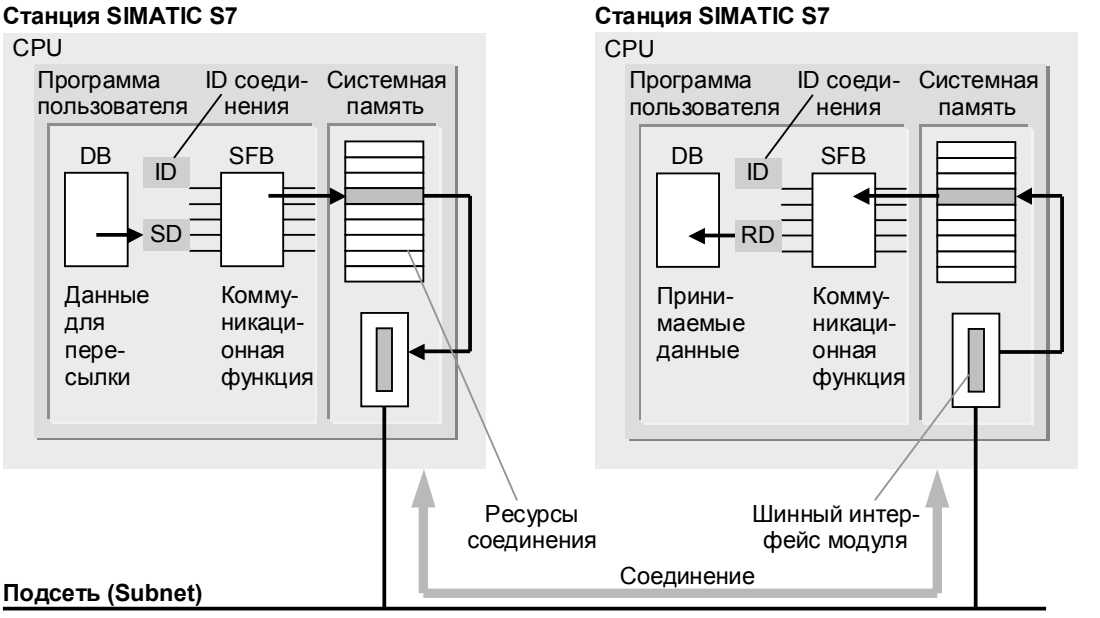


Рис. 1.5 Обмен данными между двумя станциями SIMATIC S7

Сеть

Сеть - это соединение между несколькими устройствами с целью их связи друг с другом. Она состоит из одной или нескольких идентичных или разных подсетей, связанных друг с другом.

Подсеть

В подсети все коммуникационные узлы связаны с помощью аппаратных соединений, обладающих одинаковыми физическими характеристиками и параметрами передачи, такими как скорость передачи; кроме того, обмен данными в подсети происходит в соответствии с единой процедурой передачи данных. В системе SIMATIC применяются несколько типов подсетей: MPI, PROFIBUS, Industrial Ethernet и PTP ("point-to-point" [соединение "точка к точке"]).

Служба обмена (communications service)

Служба обмена (communications service) определяет, как происходит обмен данными между коммуникационными узлами, и как эти данные обрабатываются. Служба обмена базируется на протоколе обмена, который помимо всего прочего описывает процедуру координации работы между коммуникационными узлами.

В SIMATIC различают следующие разновидности организации службы обмена: через функции S7, PROFIBUS-DP, PROFIBUS-FMS, PROFIBUS-FDL (SDA), ISO transport, ISO-on-TSP и связь через глобальные данные.

Соединение (connection)

Соединение определяет коммуникационные отношения между двумя узлами связи (коммуникационными узлами). Это есть логическое назначение двух узлов для выполнения специфических коммуникационных функций (службы обмена) и, кроме того, содержит специальные характеристики, такие как тип соединения (динамический, статический) и каким образом оно устанавливается.

В SIMATIC различают следующие разновидности соединений: S7-соединение, S7-соединение (отказоустойчивое), "point-to-point" [соединение "точка к точке"], FMS- и FDL-соединение, "ISO transport"-соединение, "ISO-on-TSP"- и TSP-соединение, UDP-соединение и E-mail-соединение.

Коммуникационные функции (communications functions)

Коммуникационные функции играют роль интерфейса между программой пользователя и службой обмена подсети. Используемые для внутренних соединений в SIMATIC S7 коммуникационные функции встроены в операционную систему CPU и вызываются с помощью системных блоков. Загружаемые блоки позволяют создавать соединение с устройствами сторонних производителей (кроме Siemens) с помощью коммуникационных процессоров.

Краткий обзор коммуникационных объектов

В таблице 1.1 показано соответствие между подсетями, службами обмена данными и модулями с функцией связи.

1.3.2 Подсети

Подсети - это часть средств связи с одинаковыми физическими характеристиками и одинаковой процедурой обработки данных. Подсети являются центральными объектами в системе связи для утилиты SIMATIC Manager.

Подсети отличаются своими рабочими характеристиками:

- MPI
  экономичный способ создания сетей для небольшого количества устройств SIMATIC с обменом малыми количествами данных.
- PROFIBUS
  высокоскоростной обмен малыми и средними объемами данных; используется прежде всего для работы с системами распределенных входов/выходов.
- Industrial Ethernet
  связь между компьютерами и PLC для высокоскоростного обмена большими объемами данных.
- PTP ("Точка к точке")
  последовательная связь между двумя коммуникационными партнерами по специальным протоколам.

Таблица 1.1 Коммуникационные объекты

Подсеть	Модули	Служба обмена данными	Конфигурация, интерфейс
MPI	Все CPU	Связь через глобальные данные (GD)	GD-таблица
		Связь между станциями посредством SFC	Вызовы SFC
		Связь посредством SFB (только при активной S7-400)	Таблица соединений, вызовы SFB
PROFIBUS	CPU с ведущим DP-устройством	PROFIBUS-DP (ведущее или ведомое устройство)	Конфигурация оборудования, входы/ выходы, вызовы SFC
		Связь внутри станции посредством SFC	Вызовы SFC
	IM 467	PROFIBUS-DP (ведущее или ведомое устройство)	Конфигурация оборудования, входы/ выходы, вызовы SFC
		Связь внутри станции посредством SFC	Вызовы SFC
	CP 342-5 CP 443-5 Extended (расширенный)	PROFIBUS-FDL, PROFIBUS-DP (ведущее или ведомое устройство)	NCM, таблица соединений, SEND / RECEIVE
		Связь внутри станции посредством SFC	Вызовы SFC
		Связь посредством SFB (только при активной S7-400)	Таблица соединений, вызовы SFB
	CP 343-5 CP 443-5 Basic (базовый)	PROFIBUS-FMS, PROFIBUS-FDL	NCM, FMS-интерфейс таблица соединений, SEND / RECEIVE
		Связь внутри станции посредством SFC	Вызовы SFC
		Связь посредством SFB (только при активной S7-400)	Таблица соединений, вызовы SFB
Industrial Ethernet	CP 343-1 CP 443-1	Транспортные протоколы ISO и TCP / IP	NCM, таблица соединений, SEND / RECEIVE
		Связь посредством SFB (только при активной S7-400)	Таблица соединений, вызовы SFB
	CP 343-1 IT CP 443-1 IT	Транспортные протоколы ISO и TCP / IP IT-связь	NCM, таблица соединений, SEND / RECEIVE
		Связь посредством SFB (только при активной S7-400)	Таблица соединений, вызовы SFB

NCM - программное обеспечение для конфигурирования CP; NCM может применяться для PROFIBUS и для Industrial Ethernet.

Посредством STEP 7 V.5 Вы можете использовать программатор для получения доступа к станциям SIMATIC S7 с помощью подсетей, чтобы, например, задать параметры или изменить программу. Шлюзы (переходы) между подсетями должны быть расположены в станции S7 с возможностью программирования.

MPI

Каждый CPU имеет интерфейс для многоточечного подключения ("multipoint interface", MPI ["многоточечный интерфейс"]). Он позволяет создать подсеть для обмена данными между CPU, PG, устройствами HMI (человеко-машинный интерфейс) согласно оригинальному протоколу обмена Siemens.

Линии передачи MPI могут иметь два типа исполнения: экранированный кабель "витая пара" или пластмассовый оптико-волоконный кабель. Длина кабеля в шинном сегменте может достигать 50 м. При этом максимальная длина может быть увеличена до 1100 м в случае применения повторителей RS485 или может даже превышать 100 км в случае применения модулей оптической связи (optical link modul). Скорость передачи данных обычно составляет 187,5 кбит/с.

Максимальное число узлов составляет 32 единицы. Каждый узел имеет доступ к шине в течение определенного отрезка времени и может в это время посылать фреймы данных. По прошествии этого промежутка времени узел передает право доступа к шине следующему узлу (процедура доступа "token passing" [передача маркера или "токена"]).

Посредством сети MPI может быть организован обмен данными между CPU с помощью установления одного из следующих типов связи: связи через глобальные данные, связи между станциями посредством SFC или связи посредством SFB. При этом не требуется применять дополнительные модули.

PROFIBUS

PROFIBUS ("**PRO**cess **FI**eld**BUS**") используется как "шина полевого уровня для автоматизации". PROFIBUS является общим стандартом, совместимым с EN 50170, для связывания в единую сеть устройств полевого уровня.

Линии передачи PROFIBUS могут иметь следующие типы исполнения: экранированный кабель "витая пара" и стеклянный или пластмассовый оптико-волоконный кабель. Максимальная длина кабеля в шинном сегменте зависит от скорости передачи данных; она может достигать 100 м при наибольшей скорости передачи (12 Мбит/с) и может достигать 1000 м при наименьшей скорости передачи (9,6 кбит/с). Длина сети может наращиваться в случае применения повторителей или модулей оптической связи (optical link modul).

Максимальное число узлов составляет 127 единицы. Различают активные и пассивные узлы. Активные узлы имеют доступ к шине в течение определенного отрезка времени и могут в это время посылать фреймы данных. По прошествии этого промежутка времени активный узел передает право доступа к шине следующему узлу (процедура доступа "token passing" [передача "токена"]). Если пассивные узлы (slaves) были назначены активному узлу (master), последний будет выполнять обмен данными с назначенными ему пассивными узлами, пока имеет доступ к шине. Пассивные узлы не получают доступа к шине.

Вы можете осуществлять связь с распределенной периферией посредством сети PROFIBUS; при этом используется соответствующая служба обмена PROFIBUS-DP. Вы можете использовать или CPU со встроенным или вставляемым ведущим DP-устройством или использовать подходящий коммуникационный процессор. В сетях PROFIBUS можно также использовать связь внутри станции посредством SFC или связь посредством SFB.

При использовании соответствующих CP возможен обмен данными посредством служб PROFIBUS-FMS и PROFIBUS-FDL. Как интерфейс для программы пользователя используются загружаемые блоки (FMS-интерфейс или SEND/RESEIVE-интерфейс).

Industrial Ethernet

Industrial Ethernet - это подсеть для обмена данными между компьютерами и программируемыми контроллерами преимущественно в промышленности в соответствии с международным стандартом IEEE 802.3.

Физически линии передачи Industrial Ethernet могут быть в виде коаксиального кабеля с двойным экранированием, в виде кабеля "витая пара" ("industrial") или в виде стеклянного оптико-волоконного кабеля. Длина электрокабеля в сети может достигать 1,5 км, тогда как длина кабеля оптической связи достигает 4,5 км. Скорость передачи данных составляет 10 Мбит/с.

Максимальное число узлов сети Industrial Ethernet может превышать 1000 единиц. Каждый узел, получающий доступ к шине, прежде всего проверяет, не посылает ли данные в это же время другой узел. Если другой узел использует в текущий момент шину, то узел, получающий доступ к шине, ожидает в течение случайным образом выбранного промежутка времени, после чего совершает новую попытку доступа к шине (процедура доступа "CSMA/CD"). Все узлы сети имеют равные права доступа.

Посредством сети Industrial Ethernet может быть также организован обмен данными с помощью установления одного из следующих типов связи: связи через S7-функции или связи посредством SFB. Если использовать для сети Industrial Ethernet соответствующие CP, то тогда есть возможность использовать связь ISO transport или ISO-on-TCP, а также использовать интерфейс SEND/RESEIVE.

Point-to-point

Интерфейс для подключения "точка к точке" ("Point-to-point", PTP) позволяет создать подсеть для обмена данными последовательной связи. Соединение "point-to-point" легко конфигурируется и обеспечивает управление как подсеть с помощью SIMATIC Manager.

Интерфейсные разъемы соединяются посредством электрокабеля. В качестве интерфейсов могут использоваться RS 232C (V.24), 20мА (TTY) и RS 422/485. Скорость передачи данных для интерфейса 20 мА составляет от 300 бит/с до 19,2 кбит/с, а для интерфейсов RS 232C и RS 422/485 - 76,8 кбит/с. Максимальная длина кабеля зависит от физического интерфейса и от скорости передачи данных; она может достигать 10 м для RS 232C, 1000 м для интерфейса 20 мА при скорости передачи 9,6 кбит/с и 1200 м для интерфейса RS 422/485 при скорости передачи 19,2 кбит/с.

3964 (R), RK 512, драйверы для принтеров и ASCII драйвер могут использоваться как протоколы (процедуры), а также последние версии пользовательских протоколов. Отдельные приложения могут потребовать использования особых драйверов.

AS-интерфейс

AS-интерфейс ("AS-Interface", AS-i) позволяет создать подсеть для обмена данными в соответствии со спецификацией IEC TG 178 для AS-интерфейса с соответствующим образом сконструированными двоичными датчиками и приводами. AS-интерфейс не рассматривается как подсеть в SIMATIC Manager; только ведущее устройство AS-i (AS-I master) может быть сконфигурировано процедурами конфигурирования аппаратной части и сети.

Линии передачи AS-Interface представляют собой неэкранированный кабель "витая пара", по которому одновременно обеспечиваются передача данных и электропитание датчиков и приводов (такая сеть требует наличия отдельного блока питания). Максимальная длина кабеля в случае применения повторителей может достигать 300 м. Скорость передачи при этом составляет 167 кбит/с.

Ведущее устройство AS-I (master) может управлять максимум 31 ведомым устройством (slave) в цикле сканирования и обеспечивая определенное время отклика.

1.3.3 Службы обмена (communications services)

Обменом данными в подсетях управляют так называемые службы обмена, тип которых определяется типом соединения. Эти службы используются преимущественно для целей, изложенных ниже:

S7-функции - это главная служба обмена в SIMATIC. S7-функции интегрированы в операционную систему CPU, и обеспечивают связь (коммуникации) между центральными процессорами, устройствами HMI и программаторами.

Ниже представлен краткий обзор их функций:

- Функции для программатора (PG):
  тестирование, запуск и сервисные функции; в PG они используются, например, для выполнения функции мониторинга переменных "monitor variables" или для чтения буфера диагностики или для запуска программ пользователя.
- Функции для человеко-машинного интерфейса (HMI):
  используется подключенными панелями оператора (OP), например, для выполнения функции чтения/записи переменных.
- SFB-коммуникации (SFB-communications):
  управляемые событиями функции для обмена большими объемами данных; запускаются вызовом SFB в программе пользователя с функциями модификации и мониторинга; статические, для сконфигурированных соединений.
- SFC-коммуникации (SFC-communications):
  управляемые событиями функции для обмена данными объемом до 76 байт за передачу; запускаются вызовом SFC в программе пользователя с функциями модификации и мониторинга; динамические, для несконфигурированных соединений.

S7-функции могут выполняться в подсетях MPI, PROFIBUS и Industrial Ethernet.

Связь через глобальные данные (Global data communications) позволяет осуществлять обмен небольшими объемами данных между несколькими CPU без дополнительного усложнения программы пользователя. Передача данных может выполняться циклически или запускаться событиями.

Связь через глобальные данные как процедура носит характер "вещания" (распространения данных); получение данных не квитируется. Состояние соединения подтверждается.

Связь через глобальные данные возможна только с MPI-шиной и K-шиной.

С **PROFIBUS-DP** осуществляется обмен данными между ведущим и ведомыми устройствами через распределенную периферию. Связь имеет "прозрачный режим" и отвечает стандарту EN 50170 том 2. С помощью данной службы обмена может быть организован доступ к ведомым устройствам, отвечающим стандартам SIMATIC S7 и прочим стандартам в подсетях PROFIBUS.

С **PROFIBUS-FMS** (Fieldbus Message Specification ["Спецификация сообщений в шине полевого уровня"]) осуществляется передача структурированных переменных (FMS-переменных) в соответствии со стандартом EN 50170 том 2. Данные коммуникации осуществляются исключительно для статических соединений в подсетях PROFIBUS.

С **PROFIBUS-FDL** (Fieldbus Data Link ["Связь через данные в шине полевого уровня"]) осуществляется передача данных с функцией SDA (Send Data with Acknowledge ["Передача данных с квитированием"]) в соответствии со стандартом EN 50170 том 2. Данные коммуникации осуществляются для статических соединений. В подсетях PROFIBUS данная служба обмена обеспечивает, например, обмен данными с контроллером SIMATIC S5.

С **ISO transport** осуществляется обмен данными в соответствии со стандартом ISO 8073 Class 4. Данные коммуникации осуществляются для статических соединений. С помощью ISO transport может быть организован, например, обмен данными с контроллером SIMATIC S5 в подсетях Industrial Ethernet.

Служба обмена **ISO-on-TSP** соответствует стандарту TCP/IP с расширением RFC 1006. Данные коммуникации осуществляются для статических соединений в подсетях Industrial Ethernet.

1.3.4 Соединения (connections)

Соединения могут быть статическими или динамическими - это зависит от выбранной службы обмена данными. Динамические соединения не конфигурируются; их установление или ликвидация определяются событиями ("Communications via non-configured connections" - "коммуникации посредством несконфигурированных соединений"). Может быть установлено только одно несконфигурированное соединение с коммуникационным партнером.

Статические соединения конфигурируются с помощью таблицы соединений (connection table). Они устанавливаются при запуске программы и остаются на все время выполнения программы ("Communications via configured connections"- "коммуникации посредством сконфигурированных соединений"). Может быть установлено несколько сконфигурированных соединений параллельно с одним коммуникационным партнером. Вы должны выбрать "Connection type" ("Тип соединения") для выбора требуемой службы обмена при конфигурировании сети (см. раздел 2.4 "Конфигурирование сети").

Вам не нужно конфигурировать соединения с помощью утилиты конфигурирования сети для служб обмена посредством глобальных данных (GD) и PROFIBUS-DP или для SFC-коммуникаций (SFC-communications) в случае обмена через S7-функции. Для обмена через GD Вы должны определить коммуникационных партнеров в таблице GD; в случае PROFIBUS-DP или SFC-коммуникаций партнеры определяются посредством адресации узлов.

Ресурсы соединения (Connection resources)

Каждое соединение требует от коммуникационного партнера-участника определенных ресурсов для конечного пункта соединения или "транзитного" пункта в модуле CP. Если, например, функции S7 выполняются посредством MPI-интерфейса CPU, соединения назначаются в CPU; такие же функции посредством MPI-интерфейса CP занимают (используют) одно соединение в CP и одно соединение в CPU.

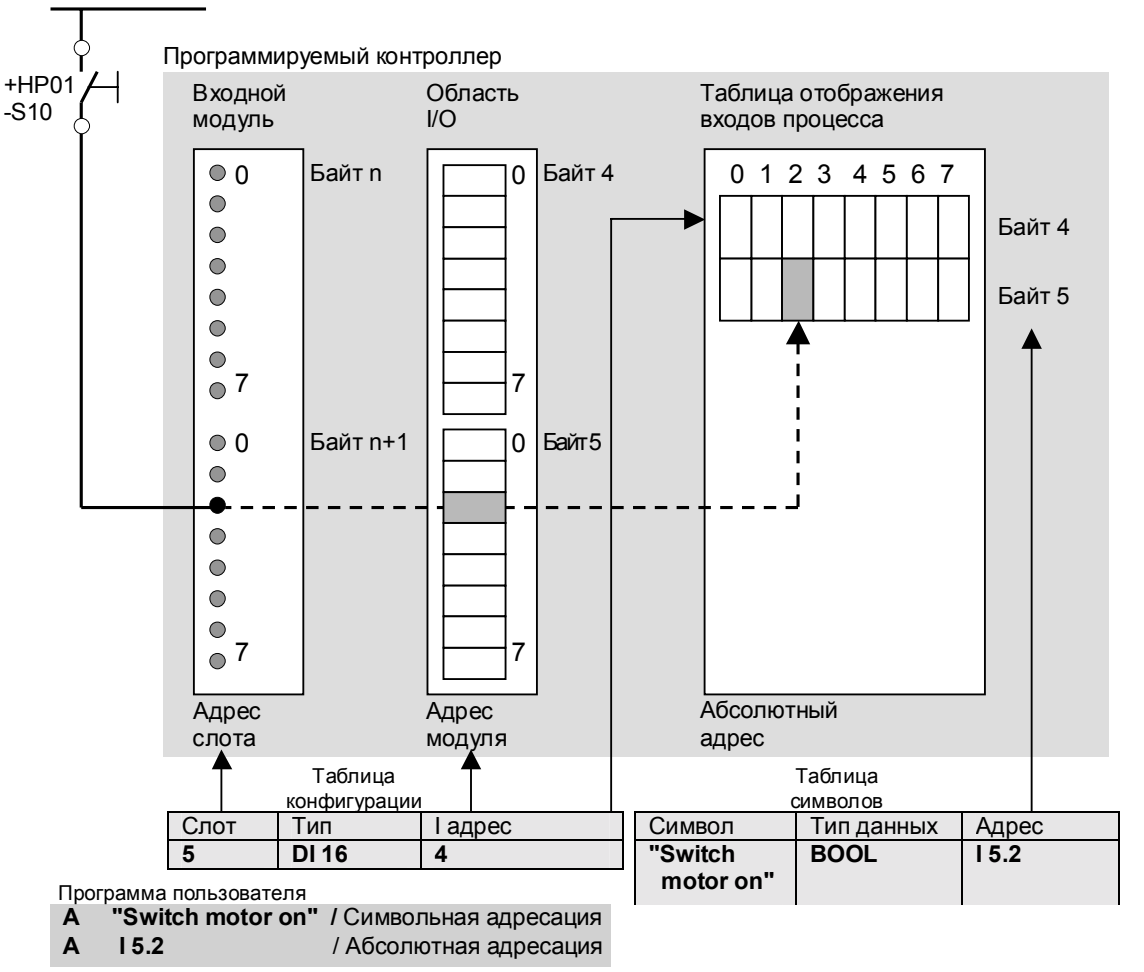
Каждый CPU имеет специальный номер для возможного соединения. Одно соединение резервируется для PG и одно соединение для OP (эти соединения не могут быть использованы для других целей).

Ресурсы соединений также требуются временно для "несконфигурированных соединений" (SFC-коммуникации).

1.4 Адресация модулей

1.4.1 Путь прохождения сигнала

При монтаже установки нужно трассировать сигналы в PLC (см. рис. 1.6).



Входной сигнал, например, сигнал от кнопки +HP01-S10, включающей мотор ("Switch motor on" - "включение мотора"), приходит во входной модуль, где он поступает на специальный терминал. Этот терминал имеет "адрес", называемый I/O-адрес (например, байт 5, бит 2).

Перед каждым началом выполнения программы CPU автоматически копирует значение сигнала в таблицу входов для сохранения "образа процесса по входу", где это значение будет доступно по входному ("input") адресу для дальнейшей обработки (например, I 5.2). При этом выражение "I 5.2" является абсолютным адресом.

Вы можете задать в таблице символов имя этому входу в виде алфавитно-цифрового символьного имени, соответствующего абсолютному адресу входного сигнала (такое, как "Switch motor on"). При этом выражение "Switch motor on" является символьным адресом.

1.4.2 Адрес слота

Каждый слот имеет фиксированный адрес в программируемом контроллере (в S7-станции). Этот адрес слота состоит из номера монтажной стойки и номера слота. Таким образом каждый модуль может быть однозначно описан указанием адреса слота ("географический адрес").

Если модуль содержит интерфейсные платы, каждая из них также описывается заданием адреса подмодуля. Таким образом, каждый дискретный или аналоговый сигнал и каждое последовательное соединение в системе имеет свой собственный уникальный адрес.

Соответственно, распределенные I/O модули также имеют "географические адреса", в данном случае включающие в себя номер системы ведущего DP-устройства и номер станции вместо номера стойки.

Вы должны использовать утилиту для конфигурирования оборудования "Hardware Configuration" для того, чтобы спланировать конфигурацию аппаратной части S7-станции, как места физического расположения модулей. Эта утилита позволяет также установить начальные адреса модулей и задать для модулей параметры (см. раздел 2.3 "Конфигурирование станций").

1.4.3 Начальный адрес модуля

Кроме адресов слотов, позволяющих определить отдельный слот, каждый модуль имеет начальный адрес, определяющий позицию в области логических адресов (область I/O-адресов). Адресное пространство входов/выходов начинается с адреса 0 и заканчивается некоторым значением, соответствующим верхней границе, которая определяется типом CPU.
Начальный адрес определяет, как адресуются входные/выходные сигналы в программе программируемого контроллера (S7-станции). Этот адрес слота состоит из номера монтажной стойки и номера слота. Таким образом каждый модуль может быть однозначно описан указанием адреса слота ("географический адрес"). В случае дискретных модулей отдельные сигналы (биты) собираются в группы по 8 (т.е. в байты). Эти байты имеют соответствующие адреса 0, 1, 2 и 3; адресация байтов начинается с начального адреса модуля. Например, в дискретном модуле с четырьмя байтами и с начальным адресов модуля 8 отдельные байты имеют адреса 8, 9, 10 и 11 соответственно.

В случае аналоговых модулей каждый из аналоговых сигналов (в виде напряжения или тока), называемых "каналами" ("channel"), занимает 2 байта. Аналоговые модули, в зависимости от конструкции имеющие 2, 4, 8 и 16 каналов, соответственно занимают 4, 8, 16 или 32 байта адресного пространства.

При включении питания (если не было предустановок) CPU устанавливает начальный адрес каждого модуля, зависящий от типа модуля, от слота и стойки. Этот начальный адрес соответствует (относительный адрес) байту 0. Вы можете видеть этот адрес в таблице конфигурации.

В системах S7-3xx с интегрированным DP-интерфейсом, S7-318 и S7-400 Вы можете изменять этот адрес. Для этого Вы должны выбрать опцию назначения начальных адресов модулей внутри разрешенного адресного пространства. Вы также можете выбрать опцию для назначения разных начальных адресов для входов и выходов для смешанных дискретных или аналоговых модулей.

Как и централизованные модули, модули распределенной периферии (станции) резервируют соответствующие номера байтов в области I/O-адресов. При этом адреса централизованных модулей и распределенных I/O не должны перекрываться.

Соответствующим образом построенные ведомые DP-устройства могут быть параметризированы таким образом, что особые номера байтов обеспечивают консистентность (логическую связанность) данных при их пересылке. Этим ведомым DP-устройствам соответствует I/O адрес одного байта, которым они адресуются при использовании системных функций SFC 14 DPRD_DAT и SFC 15 DPWR_DAT.

Дискретные модули обычно адресуются в таблицах отображения процесса таким образом, что состояния их сигналов могут автоматически обновляться и к ним обеспечивается доступ в области адресов "Input" ("Входы") и "Output" ("Выходы"). Аналоговые модули, FM и CP получают адреса не из области отображения процесса.

1.4.4 Диагностические адреса

Модули со встроенной функцией диагностики обеспечивают пользователя диагностическими данными, которые могут оцениваться в пользовательской программе. Если централизованные модули имеют адрес данных пользователя (начальный адрес модуля), то для чтения диагностических данных имеется доступ к модулю по этому адресу. Для модулей, не имеющих адреса данных пользователя, например, для источников питания, или для модулей, являющихся частью распределенной периферии, существует диагностический адрес.

Адрес данных диагностики всегда является адресом в I/O области входов и занимает один байт памяти. Длина данных пользователя по этому адресу равна 0; если данные размещены в области отображения процесса (что разрешено), то эти данные игнорируются CPU при обновлении образа процесса.

STEP 7 автоматически назначает диагностический адрес, отсчитывая длину данных вниз от верхнего максимального значения для адресов I/O. Вы можете изменять этот адрес с помощью утилиты для конфигурирования

аппаратной части.

Данные диагностики могут быть только считаны с помощью специальных системных функций; попытки доступа по адресу этих данных с помощью операторов загрузки не будут иметь эффекта (см. также раздел 20.4.1 "Адресация распределенной периферии").

1.4.5 Адреса шинных узлов

Адрес узла, номер станции

Каждая DP-станция (например, ведущее или ведомое DP-устройство или программатор) в подсети PROFIBUS имеет дополнительный адрес узла, с помощью которого станция может быть однозначно адресована на данной шине.

MPI-адрес

Модули, являющиеся узлами в MPI-сети (например, CPU, FM или CP), также имеют MPI-адрес. Этот адрес играет решающую роль для связи с PG, HMI-устройствами и для обмена посредством глобальных данных.

Нужно заметить, что в старших версиях S7-300 модули FM и CP, работающие в таких станциях, получают MPI-адрес, который выводится из MPI-адреса CPU.

В случае CPU 318 модули с MPI-связью локализуются в их собственном сегменте, так как они не имеют MPI-адреса. Они могут быть адресованы программатором посредством номера стойки и номера слота.

1.5 Адресное пространство

Адресное пространство каждого программируемого контроллера включает в себя:

- периферийные входы и выходы;
- области отображения процесса по входу и по выходу;
- области меркеров;
- области таймеров и счетчиков (см. главу 7 "Функции таймеров" и главу 8 "Функции счетчиков");
- области L-стека (см раздел 18.1.5 "Временные локальные данные").

К перечисленному нужно добавить области кода и блоков данных с локальными (внутри блока) переменными, в зависимости от программы пользователя.

1.5.1 Область данных пользователя

В SIMATIC S7 каждый модуль может иметь две области адресов: область данных пользователя, которая может быть непосредственно адресована с помощью операторов LOAD и TRANSFER и область системных данных для записей данных передачи.

При адресации модулей нет разницы в том, локализованы ли модули в стойках при централизованной конфигурации, или используются как распределенные I/O. Все модули находятся в одном и том же (логическом) адресном пространстве.

Свойства данных пользователя в модуле зависят от типа модуля. Для сигнальных модулей данные являются дискретными или аналоговыми входными/выходными сигналами. Для функциональных модулей данные могут быть, например, информацией контроля или состояния (статуса). Объем данных пользователя определяется типом модуля. Есть модули, которые располагают 1, 2, 4 или большим количеством байт в этой области. Адресация всегда начинается с байта 0. Адрес 0 байт является начальным адресом модуля; он задается в таблице конфигурации.

Данные пользователя отражает область I/O адресов, подразделяемая в зависимости от направления передачи данных на PI-область ("peripheral inputs") (область периферийных входов) и PQ-область ("peripheral outputs") (область периферийных выходов). Если данные пользователя расположены в области отображения данных процесса, то при обновлении образа процесса CPU обрабатывает данные автоматически.

Периферийные входы

Адресную область периферийных входов Вы можете использовать при чтении из области данных пользователя входных модулей. Часть PI-области адресов соответствует области отображения данных процесса. Эта часть всегда начинается с 0-го адреса I/O, при этом размер этой области определяется типом CPU.

С помощью операции прямого чтения (Direct I/O Read) Вы можете получить доступ к данным модулей, чей интерфейс не влияет на образ процесса по входу (например, аналоговые входные модули).

Состояния сигналов модулей, которые влияют на образ процесса по входу, могут также быть считаны с помощью этой операции. При этом сканируются мгновенные состояния сигналов входных битов. Необходимо учитывать, что состояния этих сигналов могут отличаться от состояний соответствующих битов области отображения входов процесса, так как образ процесса обновляется в самом начале цикла сканирования программы.

Периферийные входы могут иметь такие же абсолютные адреса, как и периферийные выходы.

Периферийные выходы

Адресную область периферийных выходов Вы можете использовать при записи в область данных пользователя выходных модулей. Часть PQ-области адресов соответствует области отображения данных процесса. Эта часть всегда начинается с 0-го адреса I/O, при этом размер этой области определяется типом CPU.

С помощью операции прямой записи (Direct I/O Write) Вы можете получить доступ к данным модулей, чей интерфейс не влияет на образ процесса по выходу (например, аналоговые выходные модули).

Состояния сигналов модулей, которые влияют на образ процесса по выходу, могут также быть изменены с помощью этой операции. При этом мгновенно изменяются состояния выходных битов. Необходимо учитывать, что изменение состояния этих битов мгновенно изменяет состояние выходных битов области отображения процесса для соответствующих модулей (!), так как нет разницы между отображением процесса по выходу и состоянием сигналов выходных модулей

Периферийные выходы могут иметь такие же абсолютные адреса, как и периферийные входы.

1.5.2 Отображение процесса (образ процесса)

Отображение процесса (образ процесса) состоит из образа дискретных входных и дискретных выходных модулей и, таким образом, подразделяется на образ входов процесса и образ выходов процесса. К образу входов процесса доступ осуществляется с помощью адресной области для входов (I), а к образу выходов процесса доступ осуществляется с помощью адресной области для выходов (Q). Как правило, установкой или процессом управление осуществляется с помощью входов и выходов. Образ процесса может быть разбит на дополнительные образы процесса, которые могут обновляться или автоматически, или под управлением пользовательской программы. Для получения более подробной информации обратитесь к разделу 20.2.1 "Обновление образа процесса".
Для S7-300 CPU и, начиная с октября 1998 г., также для S7-400 CPU Вы можете использовать адреса области отображения процесса, не занятой модулями, как дополнительную область памяти, аналогично области меркеров. Это касается как области и образа входов процесса, и образа выходов процесса.
Для отдельных CPU, скажем, для CPU 417, размер области отображения процесса может задаваться как параметр. Если Вы увеличиваете размер области отображения процесса, Вы, соответственно, уменьшаете размер рабочей (work) памяти. После изменения размера области отображения процесса CPU выполняет инициализацию рабочей (work) памяти, точно также как при холодном перезапуске.

Входы

Вход - это отображение соответствующего бита в дискретном входном модуле. Сканирование входа - это то же самое, что и сканирование бита в самом модуле. Перед выполнением программы в каждом программном цикле операционная система CPU копирует значение сигнала из модуля в образ входов процесса.
Использование образа входов процесса имеет следующие преимущества:

- Входы могут быть просканированы и записаны последовательно бит за битом (I/O биты не имеют прямого доступа).
- Сканирование входов много быстрее, чем процедура получения доступа к входному модулю (например, таким образом Вы избегаете временных потерь из-за переходных процессов в I/O шине, кроме того, время отклика системной памяти меньше, чем время отклика модуля). Следовательно, программа выполняется намного быстрее.
- Состояние входа не меняется на протяжении всего цикла программы (что означает сохранение консистентности данных на протяжении всего цикла программы). При изменении бита входного модуля это изменение состояния сигнала будет перенесено на соответствующий вход образа процесса лишь в начале следующего программного цикла.

- Входы, кроме того, могут быть установлены или сброшены, так как они находятся в RAM-памяти. С другой стороны, биты дискретных входных модулей доступны только для чтения. Входы области отображения процесса могут устанавливаться в целях отладки или запуска, для моделирования состояния датчиков. При этом заметно упрощается тестирование программы.

Эти преимущества теряются с увеличением времени отклика программы (см. раздел 20.2.4 "Время отклика").

Выходы

Выход - это отображение соответствующего бита в дискретном выходном модуле. Установка выхода - это то же самое, что и установка бита в самом модуле. Операционная система CPU копирует значение из образа выходов процесса в выходной модуль.

Использование образа выходов процесса дает следующие преимущества:

- Выходы могут быть установлены или сброшены бит за битом (прямая адресация I/O битов не возможна).
- Установка выходов много быстрее, чем процедура получения доступа к выходному модулю (например, таким образом Вы избегаете временных потерь из-за переходных процессов в I/O шине, кроме того, время отклика системной памяти меньше, чем время отклика модуля). Следовательно, программа выполняется намного быстрее.
- Состояние выхода может многократно меняться на протяжении всего цикла программы при этом состояние сигнала бита выходного модуля остается без изменения. И лишь последнее состояние сигнала будет перенесено в соответствующий выходной модуль в конце текущего программного цикла.
- Выходы, кроме того, могут быть просканированы, так как они находятся в RAM-памяти, тогда как биты дискретных выходных модулей доступны только для записи, но не для чтения. Выходы из области отображения процесса могут использоваться во всей программе.

Эти преимущества теряются с увеличением времени отклика программы. В разделе 20.2.4 "Время отклика" показано, из чего складывается время отклика программируемого контроллера.

1.5.3 Меркеры

Меркеры могут рассматриваться как дополнительные "пусковые реле" контроллера. Меркеры используются в первую очередь для хранения состояния сигналов. Они могут трактоваться как виртуальные выходы. Меркеры располагаются в области системной памяти CPU, и, следовательно, они всегда доступны. Наличное число меркеров зависит от типа выбранного CPU.

Меркеры используются для хранения промежуточных результатов, которые действительны за пределами блока, и могут обрабатываться более чем в одном блоке. Кроме блоков глобальных данных для хранения промежуточных результатов подходят:

- Временные локальные данные, возможные во всех блоках, но являющиеся действительными только для текущего вызова блока.
- Статические локальные данные, возможные только в функциональных блоках, но являющиеся действительными для многих вызовов блоков.

Ременантные меркеры

Некоторые меркеры могут быть назначены ременантными меркерами, что означает, что эти меркеры сохраняют свое состояние даже в условиях выключения питания. Ременантная область всегда начинается с 0-го адреса и заканчивается в заданном месте памяти. Ременантная область может быть задана при параметризации CPU. Более подробная информация находится в разделе 22.2.3 "Ременантность".

Тактовые меркеры

Многие процедуры в контроллере требуют периодического сигнала. Такие сигналы могут быть получены с помощью таймеров (генератор тактовых импульсов), таймерных (watchdog) прерываний (выполнение программы с управлением по времени) или просто с использованием тактовых меркеров.

Тактовые меркеры - это биты, состояния сигнала которых меняются периодически с отношением сигнал/пауза, равным 1:1. Такие биты, объединенные в байт, обеспечивают фиксированные частоты периодических колебаний (см. рис.1.7).

Вы можете задать число тактовых меркеров при параметризации CPU.

Необходимо отметить, что обновление тактовых меркеров асинхронно по отношению к выполнению главной программы.

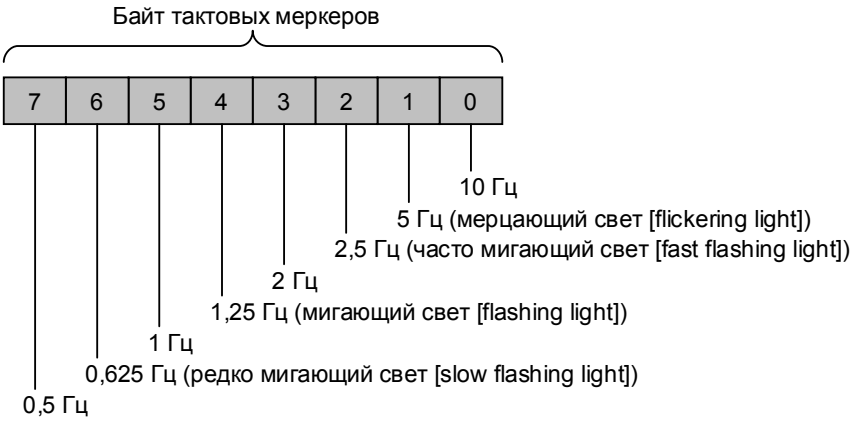


Рис.1.7 Состав байта тактовых меркеров

2 Программное обеспечение STEP 7

2.1 Базовый пакет STEP 7 (STEP 7 Basic Package)

В этой главе описывается базовый пакет STEP 7 (STEP 7 Basic Package) версии 5.1. В то время как в первой главе рассматривается краткий обзор характеристик программируемых контроллеров, в данной главе показано, как задать эти характеристики.

Базовый пакет STEP 7 (STEP 7 Basic Package) содержит следующие языки программирования: STL ("statement list" - список мнемоник), LAD ("ladder diagram" - контактный план), FBD ("function block diagram" - функциональный план). В добавление к базовому пакету возможна поставка по специальному заказу пакетов S7-SCL ("Structured Control Language" – структурированнный язык управления), S7-GRAPH (для графической разработки программ систем автоматизации SIMATIC в виде последовательности шагов и переходов между ними), S7-HiGraph (для графической разработки программ систем автоматизации SIMATIC в виде графа состояний системы и переходов между ними).

2.1.1 Инсталляция

Пакет STEP 7 V 5 является 32-разрядным приложением, работающим под управлением следующих операционных систем: Microsoft Windows 95 (начиная с сервисного пакета Service Pack 1, версии 4.00.950a), Windows 98 или Windows NT (начиная с сервисного пакета Service Pack 2, версии 4.00.1381).

Для работы с программами STEP 7 под управлением Windows 95/98 Вам потребуется программатор (PG) или компьютер (ПК) с процессором не хуже 80486 и с объемом ОЗУ не меньше 32 Мб (рекомендуемая конфигурация: процессор Pentium и объем ОЗУ 64 Мб и выше). Для работы под управлением Windows NT требуются процессор Pentium и объем ОЗУ 32 Мб и выше; кроме того необходима администраторская авторизация для инсталляции STEP 7 под Windows NT.

Если Вы работаете с большими проектами STEP 7, включающими в себя, скажем, несколько станций, состоящих из более чем 100 модулей, то Вам необходим PG или ПК с процессором повышенной производительности.

Пакет STEP 7 V 5 требует от 200 до 380 Мб памяти на жестком диске для каждой локализации (например, для англоязычной) в зависимости от операционной системы и используемой файловой системы. Также требуется предусмотреть свободное место на жестком диске для файла подкачки (от 128 до 256 Мб).

Вы должны обеспечить достаточный объем памяти на диске, содержащем данные Вашего проекта. Отдельные операции, такие как копирование проекта, могут потребовать дополнительной памяти. В случае недостаточного объема свободного пространства на диске для файла подкачки может произойти сбой в работе программы. Поэтому не рекомендуется хранить данные проекта на одном диске с файлом подкачки Windows.

Для инсталляции STEP 7 служит программа SETUP для Windows 9x/NT, которую Вы можете найти на компакт-диске. На программаторах PG STEP 7 устанавливается производителем.

Кроме STEP 7 компакт-диск также содержит программу авторизации (см. ниже), программу NCM для конфигурирования коммуникационных процессоров и электронные справочники по STEP 7 в формате Acrobat Reader V3.01.

Для интерактивного подключения требуется MPI-интерфейс. Программаторы PG имеют встроенный MPI-интерфейс, тогда как компьютеры требуют установки MPI-модуля.

Если Вам необходимо использовать модули памяти ПК, то потребуется также программатор модулей памяти.

Пакет STEP 7 V 5 имеет возможность работы в многопользовательском режиме, что означает, что проект, хранящийся на сервере, может редактироваться одновременно с нескольких рабочих станций. Для настройки Вы должны выполнить необходимые установки в панели управления Windows с помощью программы SIMATIC Workstation. В появившемся диалоговом окне Вы можете задать параметры рабочей станции для однопользовательской или многопользовательской системы с соответствующим протоколом.

2.1.2 Авторизация

Для работы с пакетом STEP 7 требуется выполнить авторизацию (подтвердить право использования). Программа авторизации находится на дискете. После инсталляции STEP 7 Вам будет предложено выполнить авторизацию, если данный жесткий диск пока не содержит авторизации. Вы также можете выполнить авторизацию позже, спустя некоторое время.

Вы можете также переносить авторизацию на другой ПК, сначала возвратив авторизацию на дискету-оригинал и затем установив ее на другом ПК.

В случае потери Вами авторизации, например, из-за выхода из строя жесткого диска, Вы можете в течение ограниченного времени (до замены Вашей авторизации) использовать "аварийную лицензию" (emergency license), находящуюся на той же дискете-оригинале (дискете с авторизацией).

2.1.3 SIMATIC Manager

SIMATIC Manager является главной утилитой STEP 7. Вы найдете ее значок на рабочем столе Windows:

SIMATIC Manager запускается двойным щелчком кнопкой мыши на значке.

При первом запуске активизируется программа "мастер проекта" (Project Wizard). Эта программа может быть использована для быстрого создания новых проектов. Тем не менее, Вы можете выключить эту программу с помощью элемента управления Check box "Display Wizard on starting the SIMATIC Manager" ("Отображать мастер-программу при запуске SIMATIC Manager"). Мастер-программа может быть вызвана при необходимости с помощью команд меню: *File (Файл) -> "New Project" Wizard*.

Процесс программирования начинается при открытии или запуске проекта ("project"). Примеры проектов представляют собой хороший материал для ознакомления.

При открытии примера проекта ZEn01_09_S7_ZEBRA с помощью команд меню: *File (Файл) -> Open (Открыть)*, Вы увидите разделенное окно проекта: слева будет структура открытого объекта (иерархическая), а справа - выбранный объект (Рис.2.1).

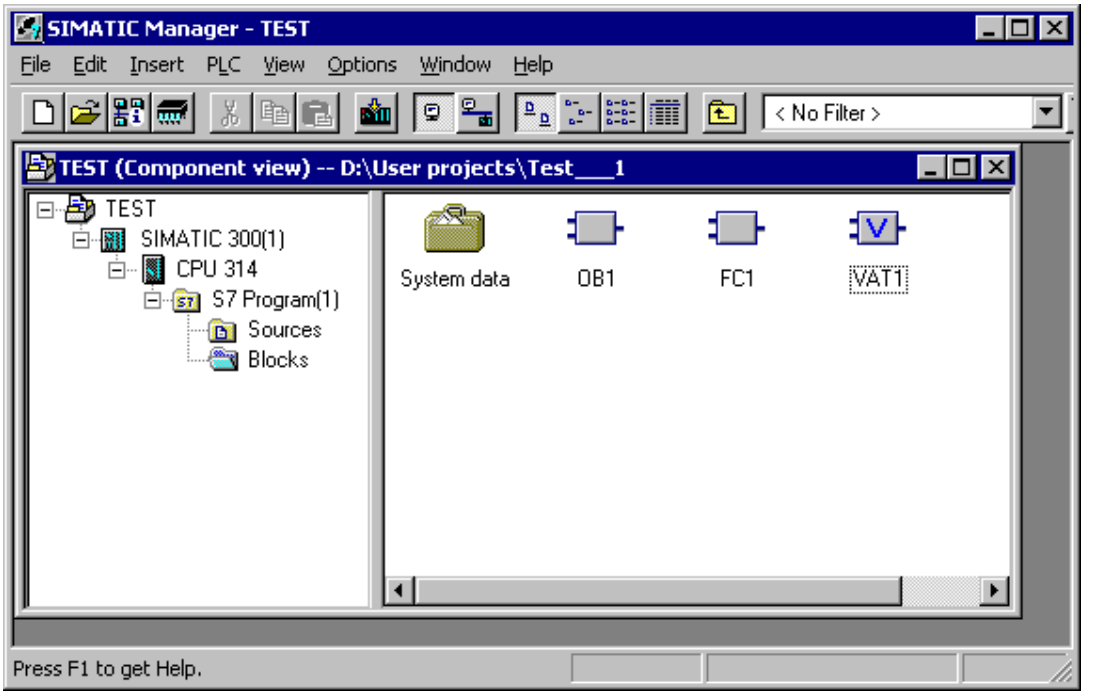


Рис.2.1 Пример открытого окна утилиты SIMATIC Manager

Щелчок на значке квадрата со знаком "+" позволяет открыть вложенные уровни структуры объекта; выбор объекта в левой части окна всегда вызывает отображение его содержания в правой части окна.

С помощью SIMATIC Manager Вы сможете работать в среде STEP 7. "Логические" объекты, отображаемые в окнах SIMATIC Manager, соответствуют "реальным" объектам Вашей установки (процесса). Проект включает в себя установку (процесс) в целом, тогда как станция (station) соответствует программируемому контроллеру (PLC).

Структура	*Объект*	*Описание*
Project	Папка для всех объектов проекта	
MPI [PTP, PROFIBUS, Ethernet]	**Subnet (Подсеть)**	Содержит параметры для подсети
SIMATIC 300/400 station	Папка для всех объектов станции	
Hardware (Аппаратное обеспечение)	**Configuration table (Таблица конфигурации)**	Содержит данные конфигурации для станции и параметры для модулей
CPU xxx	Папка для всех объектов CPU	
Connections (Соединения)	**Connection table (Таблица соединений)**	Содержит данные коммуникаций для узлов сети
S7 program	Папка для всех объектов программы пользователя	
Symbols (Символы)	**Symbol table (Таблица символов)**	Содержит символы (символьные имена) для абсолютной адресации GD
Sources	Папка для всех исходных программ	
Source files (Исходные файлы)	**Source programs (Исходные программы)**	Содержат исходные файлы программы пользователя (STL-, SCL-программы…)
Blocks	Папка для скомпилированных объектов программы пользователя	
OB n FB n FC n DB n	**Организационные блоки** **Функциональные блоки** **Функции** **Блоки данных**	Содержат скомпилированные коды и данные программы пользователя
SFC n SFB n	**Системные функции** **Системные функц. блоки**	Содержат интерфейс вызова системных блоков, встроенных в CPU
System data (Системные данные)	**Системные блоки данных**	Содержат скомпилированные данные для таблицы конфигурации
UDT n	**Data types (Пользовательские типы данных)**	Содержат определения типов данных, определенных пользователем
VAT n	**Variable table (Таблица переменных)**	Содержит переменные для мониторинга и модификации
S7 program	Папка для программы пользователя, которая назначена не для любого CPU (имеет такую же структуру, как и любая S 7-программа, назначенная CPU)	

Рис.2.2 Иерархическая структура объектов проекта STEP 7

Проект может содержать несколько станций, связанных друг с другом, например, посредством подсети MPI. Станция содержит CPU, а CPU содержит S7-программу. В свою очередь программа включает в себя другие объекты, такие как объект *Blocks* (блоки), содержащий среди прочего скомпилированные блоки.

Объекты STEP 7 объединяются в древовидную структуру. На рис. 2.2 показаны наиболее важные компоненты этой древовидной структуры ("main branch" - "главная ветвь"), которые характерны для работы с базовым пакетом S7 в автономном режиме (offline view). Объекты, выделенные жирным шрифтом, содержат другие объекты. В автономном режиме (offline view) все показанные на рисунке объекты доступны пользователю. Эти объекты расположены на жестком диске программатора PG. Если Ваш PG находится в интерактивной связи (online) с CPU (обычная система управления с PLC), Вы можете включить интерактивный режим (online view), выбрав опции меню: *View -> Online (Режим -> Интерактивный)*. Эта опция вызывает другое окно проекта, содержащее объекты назначенного устройства; при этом объекты, выделенные на рисунке, более не отображаются.

Вы можете видеть на панели заголовка окна активного проекта, работаете ли Вы в интерактивном (online) или в автономном (offline) режиме. Для более четкого разделения для панели заголовка и заголовка окна этих режимов могут быть установлены различные цвета. Для этого выберите опции меню: *Options -> Customize (Опции -> Установки пользователя)* и измените соответствующие параметры на вкладке "View" ("Режим").

Выбрав опции меню: *Options -> Customize (Опции -> Установки пользователя)*, можно изменить базовые установки SIMATIC Manager, такие как session language (язык), архив программы и место расположения для проектов, библиотек и конфигурирование архива программы.

Последовательность редактирования

Следующие пункты касаются общего редактирования объектов:

Выбрать объект - означает щелкнуть кнопкой мыши один раз на объекте в одной из частей окна проекта, после чего объект становится выделенным.

Присвоить имя объекту - означает щелкнуть кнопкой мыши на имени выбранного (см. выше) объекта, после чего вокруг имени объекта появится рамка, и Вы сможете изменить имя в окне, или, выбрав опции меню: *Edit -> Object Properties (Редактирование -> Свойства объекта)*, можно изменить имя объекта в появившемся диалоговом окне. Для некоторых объектов, таких как CPU, Вы можете изменить имя только с помощью специальных утилит (приложений), в данном случае с помощью утилиты для конфигурирования оборудования (Hardware Configuration).

Открыть объект - означает щелкнуть кнопкой мыши два раза на объекте, после чего, если объект содержит другие объекты, SIMATIC Manager отобразит его содержание в правой части окна, а если объект находится на нижнем уровне структурной иерархии, то SIMATIC Manager запустит соответствующее приложение для редактирования объекта (например, двойной щелчок на блоке запустит программу для редактирования последнего).

В данной книге пункты на стандартной панели меню в верхней части окна описаны как последовательность операторов. Программисты, имеющие опыт в использовании операторного интерфейса используют значки из панели инструментов. Использование правой кнопки мыши может быть очень полезным. Однократный щелчок правой кнопки на объекте вызывает меню с текущими опциями редактирования.

2.1.4 Проекты и библиотеки (**Project**(s) **и Library**(ies))

В STEP 7 "главные объекты", находящиеся на верхнем уровне структурной иерархии, это проекты (project) и библиотеки (library).

Проекты (projects) используются для систематического хранения данных и программ для решения задачи автоматизации. Важнейшие из них:

- данные конфигурации оборудования;
- параметры для модулей;
- данные конфигурации сетевых коммуникаций;
- программы (коды и данные, символы, исходные программы).

Объекты в проекте организованы в виде иерархической системы. Первым шагом для редактирования всех объектов проекта является открытие проекта. В следующих разделах обсуждается процесс редактирования этих объектов.

Библиотеки (library) используются для хранения многократно используемых компонентов программы. Библиотеки организованы в виде иерархической системы. Они могут содержать STEP 7 программы, которые в свою очередь могут содержать программы пользователя (скомпилированные блоки), исходные тексты программ и таблицы символов. За исключением возможности интерактивной (online) связи (не возможна отладка программы), создание программ или частей программ в библиотеке обеспечивает такие же функциональные возможности как и у объекта.

В комплекте поставки STEP 7 V5 находится стандартная библиотека Standard Library, содержащая следующие разделы:

- System Function Blocks (системные функциональные блоки), содержащие интерфейсы вызовов системных блоков для создания программ в автономном режиме, что функционально обеспечивается в CPU;
- S5-S7 Converting Blocks (блоки S5-S7 преобразования), содержащие загружаемые функции для S5-S7 преобразования (для замены стандартных функциональных блоков S5 в процессе конвертирования программы);
- T1-S7 Converting Blocks (блоки TI-S7 преобразования), содержащие дополнительные загружаемые функции и функциональные блоки для TI-S7 преобразования;
- IEC Function Blocks (функциональные блоки IEC), содержащие загружаемые функции для редактирования комплексных переменных типов DATE_AND_TIME и STRING;
- Communication Blocks (коммуникационные блоки), содержащие загружаемые функции для управления модулями CP;
- PID Control Blocks (блоки ПИД-управления), содержащий загружаемые функциональные блоки для систем автоматического управления;
- Organization Blocks (организационные блоки), содержащий шаблоны для организационных блоков (раздел объявления переменных для стартовой информации).

Вы можете найти обзор содержания этих библиотек в главе 33 "Библиотеки блоков".

Если Вы приобрели S7-модуль со стандартными блоками, программа инсталляции модуля установит эти стандартные блоки на жестком диске в виде библиотеки. В дальнейшем Вы сможете копировать эти блоки из библиотеки в Ваш проект. Библиотека может быть открыта с помощью выбора опций меню: *File -> Open (Файл -> Открыть)*, после чего она может быть отредактирована таким же образом как и проект. Вы можете также создавать свои собственные библиотеки.

С помощью выбора опций меню: *File -> New (Файл -> Создать)* может быть сгенерирован новый объект высшего уровня структурной иерархии (проект или библиотека). Место расположения вновь создаваемого объекта (проекта или библиотеки) в структуре каталогов должно быть определено с помощью опций меню: *Options -> Customize (Опции -> Установки пользователя)* или с помощью диалогового окна New (Создать).

Пункт меню Insert (Вставить) используется для добавления новых объектов в уже существующие объекты (например, для добавления нового блока в программу). Тем не менее, перед этой операцией Вы должны сначала выбрать каталог, в который Вы вставите объект из левой части окна SIMATIC Manager.

Вы можете скопировать объект с помощью опций меню: *Edit -> Copy (Правка -> Копировать)* и *Edit -> Paste (Правка -> Вставить)* или с помощью приема, распространенного среди пользователей Windows, называемого "drag-n-drop", то есть методом перетаскивания объекта с помощью мыши из одного окна в другое. Необходимо помнить, однако, что Вы не сможете отменить удаление объекта или каталога объекта в SIMATIC Manager.

2.1.5 Интерактивная справочная система (Online Help)

Интерактивная справочная система (Online Help) в SIMATIC Manager обеспечит Вас информацией в процессе программирования, что снимает необходимость пользования печатными справочными руководствами. Вы можете выбирать интересующие Вас темы, выбрав пункт меню Help (Справка). Выбор в справочной системе пункта Getting Started (Запуск), к примеру, выводит краткое резюме по использованию утилиты SIMATIC Manager.

При выборе опций: Help -> Contents (Справка -> Содержание) запускается центральная функция справочной системы STEP 7 из любого приложения. Она содержит все основные сведения.

При выборе опций: Help -> Context-Sensitive Help F1 (Справка -> Контекстная справка) запускается контекстная справочная система, то есть если Вы нажмете клавишу F1, то Вы получите информацию, соответствующую выбранному с помощью манипулятора "мышь" объекту, или информацию, соответствующую текущему сообщению об ошибке.

Если на панели Вы щелкните на кнопке со знаками стрелки и вопроса, символ вопроса добавится к указателю мыши. Установив такой указатель на объект (например, на символ или команду меню), Вы получите соответствующую интерактивную справочную информацию.

2.2 Редактирование проектов

При создании проекта Вы должны создать "каталоги" ("папки") для данных проекта, затем Вы должны сгенерировать эти данные и занести их в эти каталоги. Обычно Вы создаете проект, применяя подходящее оборудование, конфигурируете это оборудование (по крайней мере, CPU) и создаете каталог для программы пользователя. Тем не менее, Вы можете поместить S7-программу непосредственно в каталог проекта без включения какого-либо оборудования вообще. Заметьте, что инициализация модулей (изменение адресов, установки CPU, конфигурирование соединений) возможна только с помощью утилиты конфигурирования оборудования Hardware Configuration tool.

Мы настоятельно рекомендуем, чтобы редактирование проекта в целом выполнялось с использованием SIMATIC Manager. Создание, копирование или удаление каталогов или файлов, а также изменение имен (!) в структуре проекта с помощью Windows Explorer (Проводника) может привести к возникновению проблем при использовании в дальнейшем утилиты SIMATIC Manager.

2.2.1 Создание проектов

Project Wizard (Мастер проектов)

Начиная с версии STEP 7 V3.2 программа STEP 7 Wizard помогает пользователю при создании новых проектов. Пользователь должен задать тип используемого CPU, и программа-мастер создаст проект с S7-станцией и выбранным CPU, а также каталог для S7-программы, каталог для исходных программ и каталог блоков с выбранными организационными блоками.

Создание проекта с S7-станцией

Если Вы желаете создать новый проект "вручную", в данном разделе Вы найдете перечень необходимых действий, которые Вы должны будете выполнить. В разделе 2.1.3 "SIMATIC Manager" Вы найдете общую информацию по редактированию объектов.

Создание нового проекта

Выберите опции меню: *File -> New (Файл -> Создать)*, введите имя в диалоговом окне, измените тип и место расположения, если это необходимо, и подтвердите Ваш выбор щелчком на кнопке "OK" или нажатием клавиши "Enter".

Вставка новой станции в проект

Выберите проект и вставьте станцию с помощью опций меню: *Insert -> Station -> Simatic 300 Station (Вставка -> Станция -> Станция S7-300)* (в данном случае станция S7-300).

Конфигурирование станции

Щелкните на прямоугольнике со значком плюса, следующем за объектом *project* в левой части окна проекта и выберите станцию; SIMATIC Manager отображает объект Hardware (оборудование) в правой части окна. Двойным щелчком по *Hardware* запускается утилита конфигурирования оборудования Hardware Configuration, с помощью которой осуществляется редактирование таблиц конфигурации.

Если каталог модулей не показан на экране, то вызовите его с помощью опций меню: View -> Catalog (Вид -> Каталог).

Конфигурирование начинается с выбора несущей шины (rail), например, в "SIMATIC 300" и "RACK 300" и переносом методом "drag-n-drop" посредством мыши на свободное место в верхней половине окна станции (station window). При этом Вы можете наблюдать таблицу, в которой показаны слоты на шине.

На следующем этапе Вы должны выбирать требуемые модули из каталога модулей и, используя процедуру "drag-n-drop", переносить эти модули в соответствующие слоты. Для дальнейшего редактирования структуры проекта требуется установить по крайней мере один CPU, например, CPU 314 в слот 2. Вы можете добавлять остальные необходимые модули позже. Редактирование конфигурации оборудования подробно обсуждается в разделе 2.3 "Конфигурирование станций".

Затем Вы должны сохранить и скомпилировать станцию, после чего закройте ее и вернитесь в SIMATIC Manager. Кроме конфигурации оборудования открытая станция показывает также CPU.

При конфигурировании CPU утилита SIMATIC Manager также создает S7-программу со всеми объектами. Создание структуры проекта при этом завершается.

Просмотр содержания S7-программы

Откройте CPU; в правой части окна проекта Вы можете видеть символы для *S7-программы (S7-program)* и для таблицы соединений (connection table).

Откройте *S7-program*; SIMATIC Manager отображает символы для скомпилированной программы пользователя (*Blocks - Блоки*), каталог для исходных программ и таблицу символов в правой части окна.

Откройте программу пользователя (*Blocks - Блоки*); SIMATIC Manager отображает символы для скомпилированных данных конфигурации (*System data - Системные данные*) и пустой организационный блок для основной (main) программы (OB1) в правой части окна.

Редактирование объектов программы пользователя

Теперь мы достигли нижнего уровня иерархической структуры объектов. При первом открытии OB 1 отображается окно свойств объекта и запускается редактор для редактирования организационного блока. Вы можете добавлять другие пустые блоки для инкрементного редактирования посредством выбора пунктов: Insert -> S7 Block -> … (*Blocks* должно быть выделено) и выбором требуемого типа из представленного списка.

При открытии объекта *System data* (*Системные данные*) будет показан список доступных блоков системных данных. Здесь Вы получаете скомпилированные данные конфигурации.

Блоки системных данных могут быть отредактированы с помощью объекта *Hardware (Оборудование)* в каталоге *Station (Станция)*. Вы можете передать *System data (Системные данные)* в CPU, выбрав опции: *PLC -> Download (PLC -> Загрузить)*, и тем самым параметризировав CPU.

Каталог *Source Files (Исходные файлы)* пуст. Для вставки пустого исходного файла в *Source Files* Вы можете использовать меню: *Insert -> S7 Software -> STL Source File* (*Вставить -> S7ПО -> STL-исходный файл)* или для вставки в *Source Files* исходного файла, созданного в формате ASCII посредством стороннего (не из ПО STEP) редактора, Вы можете использовать меню: *Insert -> External Source File* (*Вставить -> Внешний исходный файл).*

Создание проекта без S7-станции

Если Вы желаете, Вы можете создать программу без предварительного конфигурирования станции. Для этого сами создайте каталог для программы. Выберите проект и сгенерируйте S7-программу, используя опции меню: *Insert -> Program -> S7 Program* (*Вставить -> Программу -> S7- программу)*. В данной S7-программе SIMATIC Manager создает объект *Symbols (Символы)* и каталоги объектов *Sources (Исходные файлы)* и *Blocks (Блоки).* Каталог *Blocks (Блоки)* содержит пустой блок OB 1.

Создание библиотеки

Вы можете также создать программу в объекте *library (библиотека)*, если Вы хотите использовать ее больше, чем один раз. При этом такая стандартная программа будет всегда доступна, и Вы сможете ее копировать полностью или по частям в свою текущую программу. Помните, что при этом у Вас нет возможности интерактивной (online) связи с библиотекой, и Вы сможете отладить S7-программу только в составе проекта.

2.2.2 Управление, перекомпоновка и архивирование

SIMATIC Manager поддерживает перечень всех известных "основных объектов" ("main objects"), организованных в соответствии с проектами пользователя, библиотеками и примерами (образцами) проектов. Инсталлируйте примеры (образцы) проектов со стандартными библиотеками для STEP 7 и проекты пользователя со своими собственными библиотеками.

При активации опции реорганизации *File -> Rearrange (Файл -> Реорганизация)* SIMATIC Manager убирает разрывы в пространстве памяти, получившиеся при выполнении операции удаления, оптимизируя занятое пространство памяти аналогично программе дефрагментации на жестком диске. Процесс реорганизации требует определенного времени, зависящего от объема перемещаемых данных.

Вы можете также архивировать проект или библиотеку с помощью опций: *File -> Archive (Файл -> Архивация)*. В этом случае SIMATIC Manager будет сохранять выбранный объект (каталог проекта или библиотеки со всеми подкаталогами и файлами) в сжатом виде в архивном файле.

Чтобы заархивировать проект или библиотеку, нужно использовать программу архивации. STEP 7 содержит программы архивации ARJ и PKZIP 2.50, но Вы можете использовать и другие программы архивации (например, *winzip,* начиная с версии 6.0, *pkzip,* начиная с версии 2.04g, *JAR,* начиная с версии 1.02 или *LHARC,* начиная с версии 2.13).

Проекты или библиотеки не могут быть отредактированы, когда они находятся в заархивированном (сжатом) состоянии. Вы можете "распаковывать" заархивированные объекты посредством опций: *File -> Retrieve (Файл -> Восстановление),* и после этого Вы можете редактировать эти объекты. Разархивированные объекты автоматически принимаются системой управления проектами или библиотеками.

Вы можете задавать каталоги назначения для файлов архивов и для восстанавливаемых из архивов объектов на вкладке "Archive" ("Архив"), вызываемой с помощью опций меню: *Options -> Customize (Опции -> Установки пользователя)*. Вы сами можете выбирать каталоги назначения для файлов архивов и для восстановленных файлов или можете выбрать опцию "Generate archive name automatically" ("Задавать имя архива автоматически"), что позволяет не делать никаких назначений при архивации/восстановлении, так как имя файла архива будет сгенерировано из имени проекта.

Архивирование проекта в CPU

Начиная с версии ПО STEP 7 V 5.1, при использовании соответствующих S7-400 CPU Вы можете сохранять проект в архивной (сжатой) форме в загрузочной памяти CPU, то есть в модуле памяти. Таким образом, Вы можете сохранять все данные проекта, требуемые для полностью обеспеченной обработки программы пользователя, включая таблицы символов и исходные файлы, непосредственно в установке, для которой они предназначены. Если становится необходимым модифицировать или дополнить программу, то Вы можете выгрузить эти данные на жесткий диск, сделать необходимые изменения в данных проекта и вновь сохранить обновленные данные в CPU.

При загрузке данных проекта в модуль памяти, включенный в CPU, откройте проект, отметьте CPU и выберите *PLC -> Save Project on Memory Card (PLC -> Сохранить проект в модуле памяти).* Выгрузка данных проекта из модуля памяти на жесткий диск производится при выборе: *PLC -> Retrieve Project from Memory Card (PLC -> Восстановить проект из модуля памяти).* Необходимо помнить, что при записи в модуль памяти, включенный в CPU, производится запись всего содержимого загрузочной памяти, включая системные данные и программы пользователя.

Если Вы хотите считать данные проекта, сохраненные в CPU, без создания проекта на жестком диске, то выберите соответствующий CPU с *PLC -> Display Accessible Nodes (PLC -> Отобразить доступные узлы).* Если модуль памяти включен в гнездо программатора PG, то выберите: *File -> S7 Memory Card -> Open* (*Файл -> S7 модуль памяти -> Открыть)* перед передачей данных.

2.2.3 Версии проекта (Project Versions)

Существуют три различные версии проектов SIMATIC. STEP 7 V1 позволяет создавать проекты версии 1, STEP 7 V2 позволяет создавать проекты версии 2, STEP 7 версий V3/ V4/ V5.0 позволяет создавать и редактировать проекты двух версий - 2 и 3. С помощью STEP 7 V5.1 Вы можете создавать и редактировать проекты версии 3 и библиотеки версии 3.

При наличии проекта версии 1 Вы можете преобразовать его в проект версии 2, используя опции меню: *File -> Open Version 1 Project* (*Файл -> Открыть проект версии 1)*. При этом структура проекта в программах, скомпилированные блоки версии 1, исходные STL-программы, таблица символов и конфигурация оборудования остаются неизменными.

Вы можете создавать и редактировать проекты версии 2 с помощью STEP 7 V2, V3, V4 и V5.0 (см. рис.2.3).

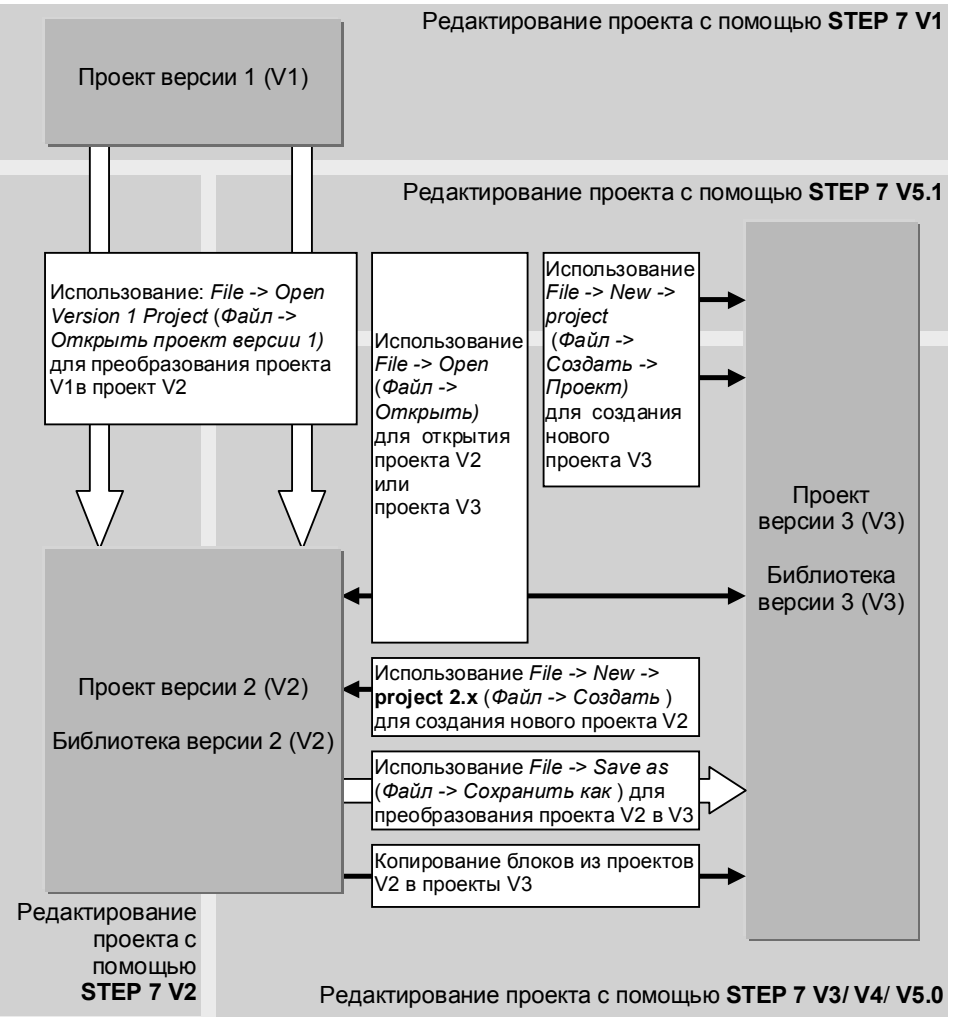


Рис.2.3 Редактирование проектов с помощью STEP различных версий

STEP 7 V5.1 позволяет работать только с проектами версии 3. Тем не менее, в этой версии Вы можете конвертировать проекты V1 в проекты V2, используя опции меню: *File -> Open Version 1 Project* (*Файл -> Открыть проект версии 1)*. Также Вы сможете открыть проект версии 2, используя опции меню: *File -> Open* (*Файл -> Открыть)*. Но при этом невозможно создавать, а также сохранять проекты в формате версии V2.

2.3 Конфигурирование станций

Для планирования конфигурации программируемого контроллера Вы должны использовать утилиту Hardware Configuration. Конфигурирование производится в автономном режиме (offline), т.е. без установления связи с CPU. Вы можете также использовать эту утилиту для адресации и параметризации модулей. Вы можете сконфигурировать оборудование на этапе планирования или же сначала инсталлировать все компоненты оборудования.

Запуск конфигурирования оборудования производится путем выбора станции с последующим выбором опций меню: *Edit -> Open Object (Правка -> Открыть объект)* или просто двойным щелчком на объекте оборудования (Hardware object) в открытом каталоге *SIMATIC 300/400 Station.* Вы можете сделать основные установки (basic settings) для оборудования, выбрав опции меню: *Options -> Customize (Опции -> Установки пользователя)*.

После выполнения конфигурирования оборудования Вы можете проверить Ваши установки на наличие ошибок с помощью выбора опций меню: *Station -> Consistency Check (Станция -> Проверка соответствия)*. При выборе опций меню: *Station -> Save (Станция -> Сохранение)* производится сохранение на жестком диске таблиц конфигурации со всеми сделанными назначениями параметров Вашего проекта.

При выборе опций меню: *Station -> Save and Compile (Станция -> Сохранение и компилирование)* производится не только сохранение на жестком диске таблиц конфигурации со всеми сделанными назначениями параметров проекта, но и их компилирование и сохранение скомпилированных данных в объекте *System data (Системные данные)* в "автономном" (offline) каталоге *Blocks (Блоки)*. После компилирования Вы можете передать сконфигурированные данные в CPU, выбрав опции меню: *PLC -> Download (PLC -> Загрузить).* Объект *System data (Системные данные)* в "интерактивном" (online) каталоге *Blocks (Блоки)* содержит текущие данные конфигурации в CPU. Вы можете "вернуть" эти данные на жесткий диск, выбрав опции меню: *PLC -> Upload (PLC -> Выгрузить).*

Вы можете также экспортировать данные конфигурирования оборудования, выбрав опции меню: *Station -> Export (Станция -> Экспорт)*. В этом случае STEP 7 создаст файл в ASCII формате, который будет содержать данные конфигурации и данные параметризации модулей. При этом Вы можете выбирать между текстовым форматом файла, когда данные можно прочитать в виде английских символов, и компактным (шестнадцатеричным) форматом данных. Вы можете также импортировать соответствующим образом структурированный ASCII файл.

Контрольная сумма (Checksum)

Утилита для конфигурирования оборудования Hardware Configuration генерирует контрольную сумму для корректно скомпилированной станции и сохраняет ее в системных данных. Идентичная системная конфигурация будет иметь точно такую же контрольную сумму, поэтому Вы можете легко сравнивать "автономную" (offline) и "интерактивную" (online) конфигурации.

Собственно контрольная сумма (Checksum) является характеристикой объекта *System data (Системные данные).*

Для считывания контрольной суммы откройте каталог *Blocks (Блоки)* в S7-программе, выберите объект *System data (Системные данные)* и откройте его с помощью опций: *Edit -> Open Object (Правка -> Открыть объект).* Программа пользователя также имеет собственное значение контрольной суммы. Вы можете найти этот параметр среди контрольных сумм системных данных в свойствах *Blocks (Блоки)*: выберите каталог *Blocks (Блоки)*, а затем опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)* на вкладке "Checksums" (контрольные суммы).

Окно станции (Station)

При открытии утилиты для конфигурирования оборудования Hardware Configuration отображается окно станции и каталог оборудования (см. ниже рис. 2.4).

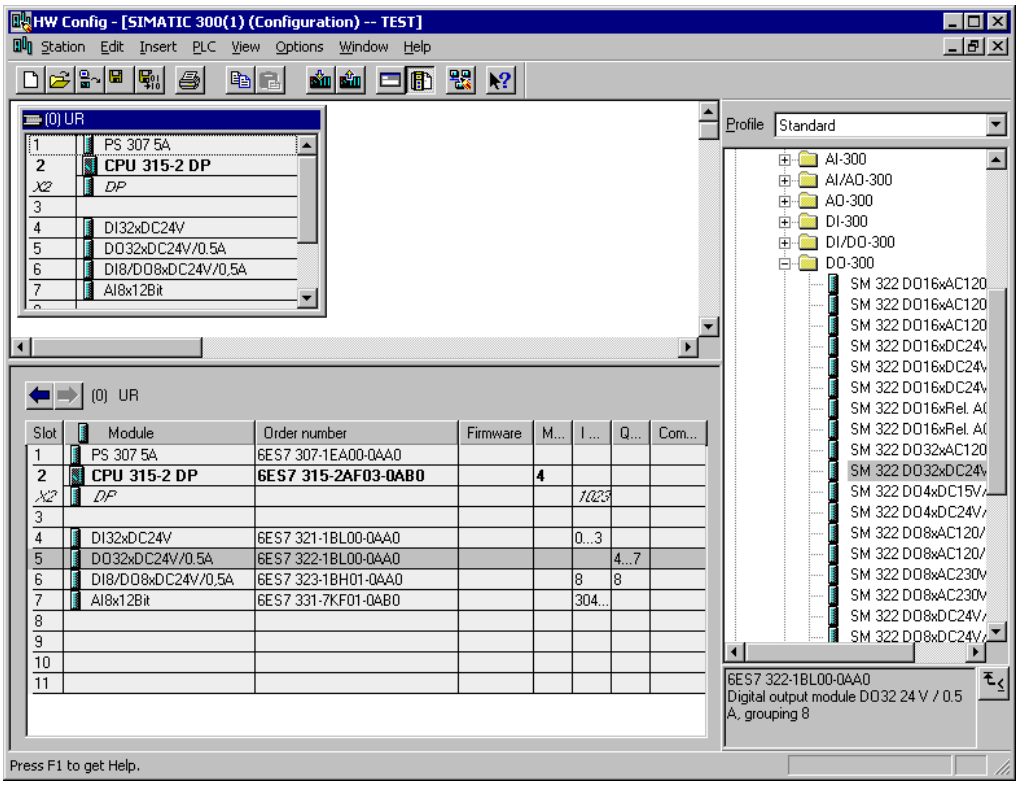


Рис. 2.4 Пример окна станции, открытого утилитой конфигурирования оборудования Hardware Configuration

Для удобства редактирования Вы можете увеличить (максимизировать) окно станции. В верхней части окна показаны монтажные стойки в форме таблиц и DP-станции в форме символов. Если используются несколько стоек, то Вы можете видеть соединения между интерфейсными модулями, а если используется подсеть PROFIBUS, Вы можете видеть систему ведущего DP-устройства. В нижней части окна станции показана таблица конфигурации, которая дает подробную информацию о стойке или о ведомом DP-устройстве, выбранном в верхней части окна.

Каталог оборудования (Hardware)

Вы можете скрывать и показывать каталог оборудования с помощью опций: *View -> Catalog (Вид -> Каталог).* Каталог отображает все доступные монтажные стойки, модули, интерфейсные модули, совместимые с STEP 7. С помощью опций: *Options -> Edit Catalog Profile (Опции -> Редактирование профиля каталога)* Вы можете скомпилировать свой собственный каталог оборудования, который будет отображать только тот набор модулей, который используется Вами в структурах контроллеров. Двойным щелчком на панели заголовка Вы можете "пристыковать" каталог к правому краю окна станции или вновь освободить его.

Таблица конфигурации (Configuration table)

Утилита для конфигурирования оборудования Hardware Configuration работает с таблицами, каждая из которых представляет монтажную стойку, модуль или DP-станцию. Таблица конфигурации показывает слоты с модулями, установленными в них или свойства модулей, такие как адреса или порядковые номера. Двойной щелчок на строке модуля открывает окно свойств модуля (properties), с помощью этого окна можно задать параметры модуля.

2.3.1 Конфигурирование модулей

Конфигурирование начинается с выбора и переноса с помощью манипулятора "мышь" упомянутым выше методом "drag-n-drop" монтажной шины из каталога, например, "SIMATIC 300" или "RACK 300" в верхнюю половину окна станции. Пустая таблица конфигурации для центральной стойки отображается в окне. Теперь выберите требуемые модули из каталога модулей и перенесите тем же способом в подходящие слоты. Символ, говорящий о невозможности назначения данного слота "No Parking" ("Нет установки") для выбранного модуля, появится при попытке назначения уже назначенного слота.

В случае однорядной S7-300 станции слот 3 оставляется пустым: он резервируется для интерфейсного модуля для связи со стойкой расширения.

Вы можете сгенерировать таблицу конфигурации для другой стойки переносом с помощью мыши выбранной монтажной стойки из каталога в окно станции. В системах S7-400 несоединенной стойке (более точно: соответствующему приемному интерфейсному модулю) назначается интерфейс с помощью вкладки "Link" (соединение) в окне свойств ("Properties") передающего ("Send") IM. Для этого выберите модуль, затем опции меню: *Edit -> Object Properties (Правка -> Свойства объекта*).

Конфигурирование станций распределенных I/O модулей описано в главе 20.4.2 "Конфигурирование распределенных I/O".

2.3.2 Адресация модулей

При конфигурировании модулей утилита конфигурирования оборудования Hardware Configuration автоматически назначает начальные адреса модулей.

Вы можете видеть эти адреса в нижней части окна станции в свойствах объекта для соответствующих модулей. Для S7-400 CPU и S7-300 CPU с встроенным DP-интерфейсом Вы можете изменять адреса модулей. При этом необходимо учитывать правила адресации для систем S7-400 и S7-300, также как и диапазоны адресации для отдельных модулей.

Существуют модули, имеющие и входы и выходы, для которых Вы можете (теоретически) резервировать различные начальные адреса. При этом необходимо учитывать специальную информацию, предлагаемую в руководствах по использованию этих изделий; подавляющее большинство функциональных и коммуникационных модулей требуют использовать одинаковые начальные адреса для входов и выходов.

При назначении начальных адресов модулям для системы S7-400, Вы также можете выполнить назначение для дополнительного образа процесса. Если в центральной стойке используется более чем один процессор, то автоматически устанавливается мультипроцессорный режим, и Вы должны назначить модуль для CPU.

С помощью опций: *View -> Address Overview (Вид -> Обзор адресов)* в появившемся окне Вы можете получить информацию о текущей адресации всех модулей для выбранного CPU.

Модули на шине MPI или на коммуникационных шинах имеют MPI-адрес. Вы можете изменять этот адрес. Тем не менее, необходимо помнить, что новые MPI-адреса вступят в силу только после того, как данные конфигурации будут пересланы в CPU.

Символы для адресов, назначенных пользователем

Вы можете использовать утилиту для конфигурирования оборудования Hardware Configuration для назначения символов (имен) входам и выходам, которые заносятся в таблицу символов (Symbol Table).

После конфигурирования и адресации дискретных и аналоговых модулей Вы должны сохранить данные станции. После этого Вы должны выбрать модуль (строку в таблице) и с помощью опций: *Edit -> Symbols (Правка -> Символы)* открыть окно, в котором Вы сможете назначить символы, типы данных и комментарии к абсолютному адресу для каждого канала (побитно для дискретных модулей и пословно для аналоговых модулей).

Кнопка "Add Symbol" ("Назначить символ") служит для замены абсолютной адресации без символов на абсолютную адресацию с символами. Кнопка "Apply" ("Применить") позволяет занести символы в таблицу символов (Symbol Table). Кнопкой "OK" закрывают окно диалога.

2.3.3 Параметризация модулей

При назначении параметров модулям определяются их свойства. Задавать параметры модуля необходимо, только когда Вы хотите изменить параметры, заданные по умолчанию. Для параметризации модуля требуется, чтобы он был в таблице конфигурации.

Для редактирования свойства модуля открываются или двойным щелчком на модуле в таблице конфигурации, или выбором в таблице строки с модулем с

последующим выбором опций меню: *Edit -> Object Properties (Правка -> Свойства объекта*). В раскрывшемся диалоговом окне Вы найдете несколько вкладок с определяемыми параметрами. При использовании этого метода для параметризации CPU Вы можете задавать характеристики выполнения Вашей программы пользователя.

Некоторые модули позволяют устанавливать их параметры в процессе выполнения программы с помощью программы пользователя посредством системных функций SFC 55 WR_PARM, SFC 56 WR_DPARM и SFC 57 PARM_MOD.

2.3.4 Объединение в сеть модулей посредством MPI

Вы должны определить узлы для MPI-подсети в свойствах модулей (Module Properties). Для этого выберите в таблице конфигурации CPU или интерфейсную плату MPI, если она установлена, и откройте свойства устройства, используя опции меню: *Edit -> Object Properties (Правка -> Свойства объекта*). В появившемся окне диалога на вкладке "General" ("Общие") Вы найдете кнопку "Properties" (свойства) на панели "Interface" (интерфейс). Щелкнув на этой кнопке, Вы откроете другое окно диалога с вкладкой "Parameter" (параметр), где расположены данные подсети.

Таким образом удобно также задавать MPI-адрес, который Вы должны установить для данного CPU. Надо заметить, что для старших S7-300 CPU, модулей FM и CP MPI-адрес для MPI-соединения задается автоматически, исходя из CPU.

Старший MPI-адрес должен быть больше или равным старшему MPI-адресу, назначенному в подсети (примите во внимание автоматическое назначение для FM и CP!). Он должен иметь одинаковое значение для всех узлов подсети.

Совет: если у Вас имеется несколько станций с CPU одинакового типа, то для CPU в разных станциях назначьте различные имена (идентификаторы). По умолчанию они все имеют имя "CPUxxx(1)", так что в подсети они могут различаться только по их MPI-адресам. Если Вы не желаете сами назначать имена для CPU, Вы можете изменить номера в скобках, т.е. изменить имя "CPUxxx(1)" на имя "CPUxxx(n)", где "n" равно MPI-адресу.

При назначении MPI-адреса примите во внимание возможность подключения в дальнейшем к MPI-сети программатора PG или панели оператора (OP) для целей управления и технического обслуживания. Вы должны подключить постоянно установленные программаторы PG или панели оператора (OP) непосредственно к MPI-сети; для подключения устройств с помощью отвода (spur-line) предназначен специальный разъем - MPI-коннектор с резьбовым соединением.

Совет: зарезервируйте адрес 0 для программатора обслуживания, адрес 1 - для сервисной панели OP и адрес 3 - для сменного CPU (в соответствии с адресацией, принятой по умолчанию).

2.3.5 Режимы Monitor (мониторинг) и Modify (обновление) в модулях

С помощью утилиты для конфигурирования оборудования Hardware Configuration Вы можете выполнить проверку монтажа установки без программы пользователя. Для этого требуется, чтобы программатор был интерактивно (online) подключен к станции и конфигурация была сохранена, скомпилирована и загружена в CPU. После выполнения выше указанных условий Вы можете вызывать каждый дискретный и аналоговый модуль. Выбрав модуль, с помощью опций меню: *PLC -> Monitor/Modify (PLC -> Мониторинг/Обновление*) установите соответствующие режимы работы и условия запуска.

С помощью кнопки "Status Value" (значение состояния) утилита для конфигурирования оборудования Hardware Configuration покажет Вам состояние сигналов или каналов модулей. С помощью кнопки "Modify Value" (измененное значение) производится запись в модуль значений, записанных в одноименной колонке "Modify Value".

Если активен элемент управления checkbox "I/O Display" (отображение I/O), то вместо входов/выходов образа процесса будут отображаться состояние периферийных входов/выходов (память модуля). Если активен элемент управления checkbox "Enable Periph Outputs" (Разблокировка периферийных выходов), то отменяется блокировка выходных модулей, если CPU находится в режиме STOP (см. раздел 2.7.5 "Разблокировка периферийных выходов").

Вы можете найти и другие способы мониторинга и обновления входов и выходов в разделах 2.7.3 "Разблокировка периферийных выходов" и 2.7.4 "изменение переменной".

2.4 Конфигурирование сети (Network)

Основой коммуникаций в SIMATIC является объединение в сеть S7-станций. Для организации сети требуется наличие подсетей и модулей с коммуникационными свойствами в станциях. Вы можете создавать подсети и станции внутри иерархической структуры проекта посредством утилиты SIMATIC Manager. После этого Вы можете добавлять модули с коммуникационными свойствами (такие как CPU и CP), используя утилиту для конфигурирования оборудования Hardware Configuration; одновременно Вы можете назначать подсети коммуникационные интерфейсы этих модулей. Затем Вы можете установить коммуникационные отношения - соединения (connection) между этими модулями посредством утилиты конфигурирования сети Network Configuration в таблице соединений (connection table).

Утилита конфигурирования сети Network Configuration позволяет графически представить и документировать сконфигурированные сети и их узлы. С помощью утилиты Network Configuration Вы можете также создать все необходимые подсети и станции; затем Вы должны назначить станции в подсетях и параметризировать свойства узлов ("node properties") для модулей с коммуникационными свойствами.

Для установления соединений (connections) посредством утилиты конфигурирования сети Network Configuration Вы можете действовать по следующему плану:

- Откройте объект MPI-подсеть, созданную стандартным способом в каталоге проекта. Если она отсутствует, просто создайте новую подсеть с помощью опций: *Insert -> Subnet (Вставка -> Подсеть)*.
- С помощью утилиты конфигурирования сети Network Configuration создайте необходимые станции и, если требуется, другие подсети.
- Откройте объекты station (станции) и снабдите их модулями с коммуникационными свойствами.
- Соедините модули посредством определенных подсетей.
- Настройте параметры сети, если это необходимо.
- Задайте коммуникационные соединения (communication connections) в таблице соединений (connection table), если это необходимо.

Вы можете также сконфигурировать связь через глобальные данные с помощью утилиты конфигурирования сети Network Configuration: выберите подсеть MPI и затем опции: *Options -> Define Global Data (Опции -> Определить глобальные данные)* (см. раздел 20.5 "Связь посредством глобальных данных").

Опции меню: *Network -> Save (Сеть -> Сохранить)* позволяют сохранить промежуточные результаты конфигурирования сети. Вы можете проверить корректность конфигурации сети посредством опций меню: *Network -> Consistency Check (Сеть -> Проверка корректности).*

Закрытие процесса конфигурирования сети производится посредством выбора опций: *Network -> Save and Compile (Сеть -> Сохранить и скомпилировать).*

Окно Network (Сеть)

Чтобы запустить утилиту конфигурирования сети Network Configuration Вы должны создать проект. Вместе с проектом утилита SIMATIC Manager автоматически создает MPI-подсеть.

Двойным щелчком на этом объекте или на любой другой подсети запускается утилита конфигурирования сети Network Configuration. Вы также можете запустить эту утилиту, если откроете объект *Connections (Соединения)* в каталоге *CPU*.

Ниже на рис. 2.5 представлено окно утилиты конфигурирования сети Network Configuration, отображающее все ранее созданные подсети и станции (узлы) проекта с сконфигурированными соединениями (connections).

Таблица соединений (connection table) показана в нижней части окна. Она появляется, если в верхней части окна выделен модуль, обладающий коммуникационными свойствами, например, S7-400 CPU.

Второе окно отображает каталог объектов сети с выбранными SIMATIC станциями, подсетями и DP-станциями. Вы можете скрыть каталог или вновь открыть его с помощью опций: *View -> Catalog (Вид -> Каталог).* Двойным щелчком на панели заголовка Вы можете "пристыковать" каталог к правому краю окна станции или вновь освободить его.

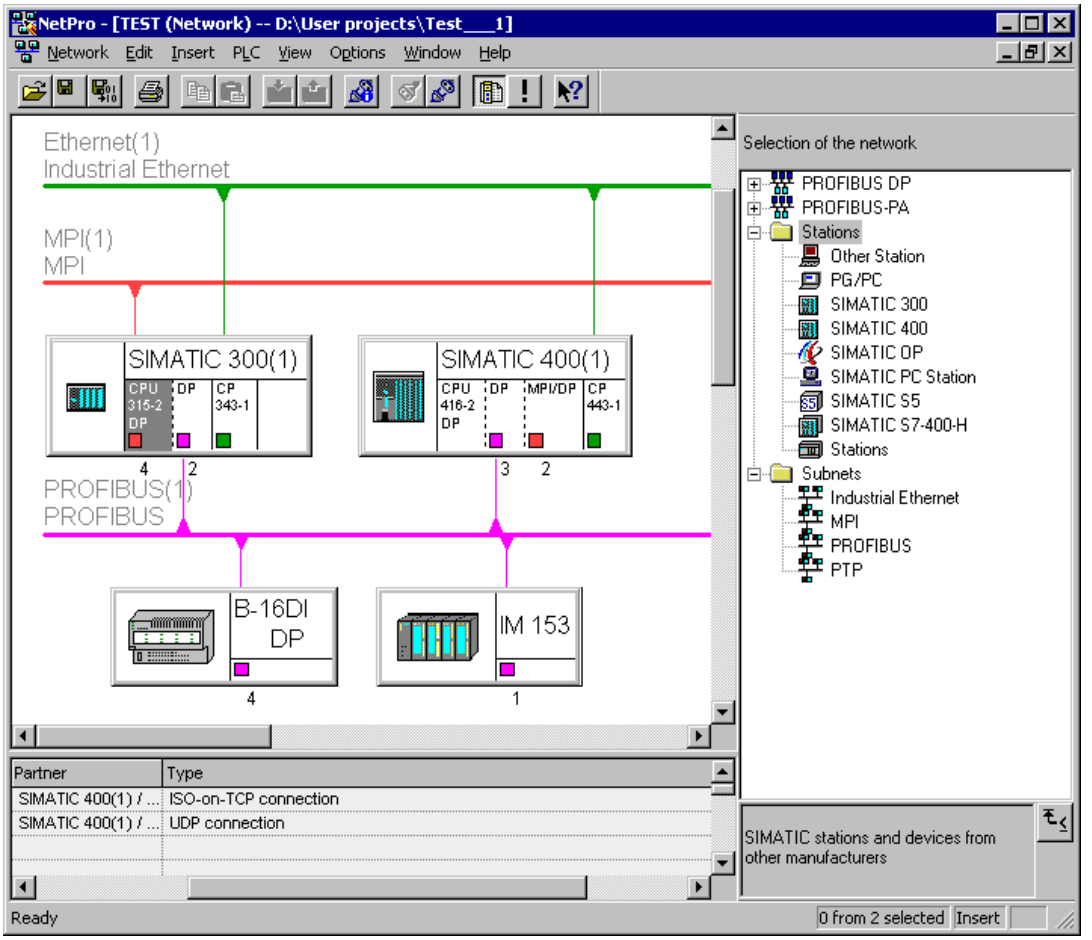


Рис. 2.5 Пример окна утилиты конфигурирования сети Network Configuration

С помощью опций: *View -> Zoom In (Вид -> Увеличить масштаб), View -> Zoom Out (Вид -> Уменьшить масштаб)* и *View -> Zoom Factor (Вид -> Коэффициент масштабирования)* Вы можете настраивать четкость графического представления конфигурации сети.

2.4.1 Конфигурирование графического представления сети (Network View)

Выбор и монтаж компонентов

Конфигурирование сети начинается с выбора типа подсети с помощью манипулятора "мышь" из каталога и переноса этого объекта в окно сети. Подсеть в этом окне представляется горизонтальной линией. Запрещенные позиции для нее отображаются рядом с указателем в виде знака запрета.

Тем же способом Вы должны выбрать и установить станции, сначала без связи с подсетью. Сначала станции "пустые". Двойной щелчок на станции запускает утилиту конфигурирования оборудования Hardware configuration, с помощью которой можно сконфигурировать станцию или, по крайней мере, модуль (модули) для объединения в сеть. После этого необходимо сохранить станцию и вернуться к конфигурированию сети (Network Configuration).

Интерфейс модуля, обладающего коммуникационными свойствами, отображается в окне утилиты конфигурирования сети, как маленький прямоугольник под изображением модуля. Щелкните на нем, удержите кнопку мыши и "перетащите" к требуемой подсети. Соединение с подсетью выглядит на схеме как вертикальная линия.

Выполните такие же операции со всеми другими узлами.

Вы можете перемещать созданные подсети и станции в окне сети. Таким способом Вы можете представить конфигурацию Вашего оборудования визуально.

Установка коммуникационных свойств

После создания графического представления сети, Вы должны параметризировать подсети. Для этого выберите подсети и с помощью опций меню: *Edit -> Object Properties (Правка -> Свойства объекта*) откройте окно свойств. На вкладке окна "General" (Общие) находится идентификатор S7-подсети (ID). ID состоит из двух шестнадцатеричных чисел - номера проекта и номера подсети. Данный ID S7-подсети необходим при переходе в интерактивный режим (online) с программатором без соответствующего проекта, чтобы подключиться к другим узлам посредством подсети. Вы можете установить свойства сети (network properties) на вкладке "Network Settings" ("установки сети"), например, скорость передачи данных (data transfer rate) или старший адрес узла (highest node address).

Выбрав для узла подключение к сети (network connection), Вы можете определить свойства сети с помощью опций меню: *Edit -> Object Properties (Правка -> Свойства объекта)*, например, адрес узла (node address) и подсеть (subnet), к которой он подключен, или же Вы можете создать новую подсеть.

На вкладке "Interfaces" ("Интерфейсы") окна свойств станции Вы можете видеть все модули с коммуникационными свойствами с адресами узлов и используемыми типами подсетей.

Аналогичным образом Вы можете определять свойства модулей узлов (задавая параметры в окнах ввода утилиты для конфигурирования оборудования Hardware Configuration).

2.4.2 Конфигурирование системы ведущего DP-устройства с помощью утилиты конфигурирования сети Network Configuration

Вы можете также использовать утилиту конфигурирования сети Network Configuration для конфигурирования системы распределенных I/O. Используя

опции меню: *View -> with DP Slaves (Вид -> с ведомыми DP-устройствами),* Вы можете отобразить или скрыть изображение ведомых (slave) DP-устройств в графическом представлении сети (Network View).

Для конфигурирования системы ведущего DP-устройства Вам требуются:

- Подсеть PROFIBUS (если подсеть отсутствует, "перетащите" объект подсеть PROFIBUS в окно сети из каталога объектов сети).
- Ведущее DP-устройство (master) в станции (если устройство отсутствует, "перетащите" станцию в окно сети из каталога объектов сети, откройте станцию и выберите DP-устройство с помощью утилиты для конфигурирования оборудования Hardware Configuration либо как встроенное в CPU устройство, либо как автономный модуль).
- Соединение (connection) для DP-устройства с подсетью PROFIBUS (или выделите подсеть с помощью утилиты для конфигурирования оборудования Hardware Configuration, или щелкните кнопкой мыши на соединении для ведущего DP-устройства (master) при использовании утилиты для конфигурирования сети Network Configuration и "перетащите" объект на подсеть PROFIBUS).

В окне сети выберите ведущее DP-устройство (master), которому должно быть назначено ведомое DP-устройство (slave). Найдите ведомое DP-устройство (slave) в каталоге объектов сети в "PROFIBUS" в соответствующем подкаталоге, "перетащите" объект в окно сети и задайте свойства в появившемся окне.

Параметризируйте ведомое DP-устройство (slave) сначала выделив его и далее используя опции меню: *Edit -> Open Object (Правка -> Открыть объект)*. После этого будет запущена утилита для конфигурирования оборудования Hardware Configuration. Теперь Вы можете выполнить адресацию данных пользователя или, в случае использования модульных ведомых устройств (slave), выберите I/O модули (см. раздел 2.3 "Конфигурирование станций").

Вы сможете подключить интеллектуальное ведомое DP-устройство к подсети, только если Вы предварительно создали его (см. раздел 20.4.2 "Конфигурирование распределенных I/O"). В каталоге объектов сети типы интеллектуальных DP-устройств (slave) находятся под "Already created stations" ("Готовые станции"). При выбранном ведущем DP-устройстве (master) Вы можете "перетащить" объект в окно сети и задать свойства в появившемся окне свойств объекта (как в утилите для конфигурирования оборудования Hardware Configuration).

С помощью выбора опций меню: *View -> Highlight -> Master System (Вид -> Выделить -> Система ведущего DP-устройства)* Вы можете выделять назначения узлов системы ведущего DP-устройства (DP-master system). При этом Вы должны сначала выбрать (выделить) ведущее (master) или ведомое (slave) устройство этой системы.

2.4.3 Конфигурирование соединений (Connections)

Соединение (Connection) описывает коммуникационные отношения между двумя устройствами. Соединения должны быть сконфигурированы

- если Вы хотите установить SFB-коммуникации между двумя SIMATIC S7-устройствами ("Communications via configured connections" - "коммуникации посредством сконфигурированных соединений") или
- если коммуникационный партнер не является SIMATIC S7-устройством.

Примечание: Вам нет необходимости конфигурировать соединение для прямого интерактивного (online) соединения программатора с MPI-сетью в целях программирования или отладки. Если же необходимо установить связь программатора с другими узлами, скомпонованными в других связанных подсетях, Вы должны будете сконфигурировать соединение программатора. Для этого двойным щелчком выберите в каталоге объектов сети (Network Object Catalog) объект PG/PC в каталоге Stations (Станции), откройте PG/PC двойным щелчком в окне сети (network), выберите интерфейс и назначьте его для подсети.

Connection table (таблица соединений)

Коммуникационные соединения конфигурируются в таблице соединений (Connection table).

Требование: Вы должны создать проект со всеми станциями, которые могут обмениваться данными друг с другом, также Вы должны назначить модули с коммуникационными свойствами для подсети.

Объект *Connections (Соединения)* в каталоге *CPU* представляет таблицу соединений (Connection table). Двойной щелчок на объекте *Connections (Соединения)* запускает утилиту конфигурирования сети Network Configuration, также как и двойной щелчок на подсети в каталоге проекта.

Для конфигурирования соединений выберите S7-400 CPU в утилите конфигурирования сети Network Configuration. В нижней части окна сети располагается таблица соединений (Connection table) (см. пример: таблица 2.1). Если таблица не видна, то поместите указатель мыши на нижний край окна и, когда изменится его форма, перемещайте край окна вверх.

Таблица 2.1 Пример таблицы соединений (Connection table)

Local ID (локальный ID)	Partner ID (ID партнера)	Partner (партнер)	Type (тип)	Active Connection Buildup (активация соединения)	Send Operating State Message (посылать сообщение о рабочем состоянии)
1	1	Station 416/CPU416(5)	S7 connection	Yes (да)	No (нет)
2	2	Station 416/CPU416(5)	S7 connection	Yes (да)	No (нет)
3		Station 315/CPU315(7)	S7 connection	Yes (да)	No (нет)
4	1	Station 417/CPU414(4)	S7 connection	Yes (да)	No (нет)

Вы можете вводить новые коммуникационные соединения с помощью опций: *Insert -> New Connections (Вставка -> Новое соединение)* или дважды щелкнув на пустой строке.

Вы должны создавать соединение (connection) для каждого активного ("active") CPU. Надо заметить, что для S7-300 CPU Вы не сможете создать таблицу соединений; S7-300 CPU могут быть только "пассивными" ("passive") партнерами в S7-соединениях.

В окне "New Connection" ("Новое соединение") Вы можете выбирать коммуникационных партнеров в диалоговых окнах "Station" ("Станция") и "Module" ("Модуль") (см. рис. 2.6); выбираемые станция или модуль при этом должны уже существовать. Вы также можете определять тип соединения.

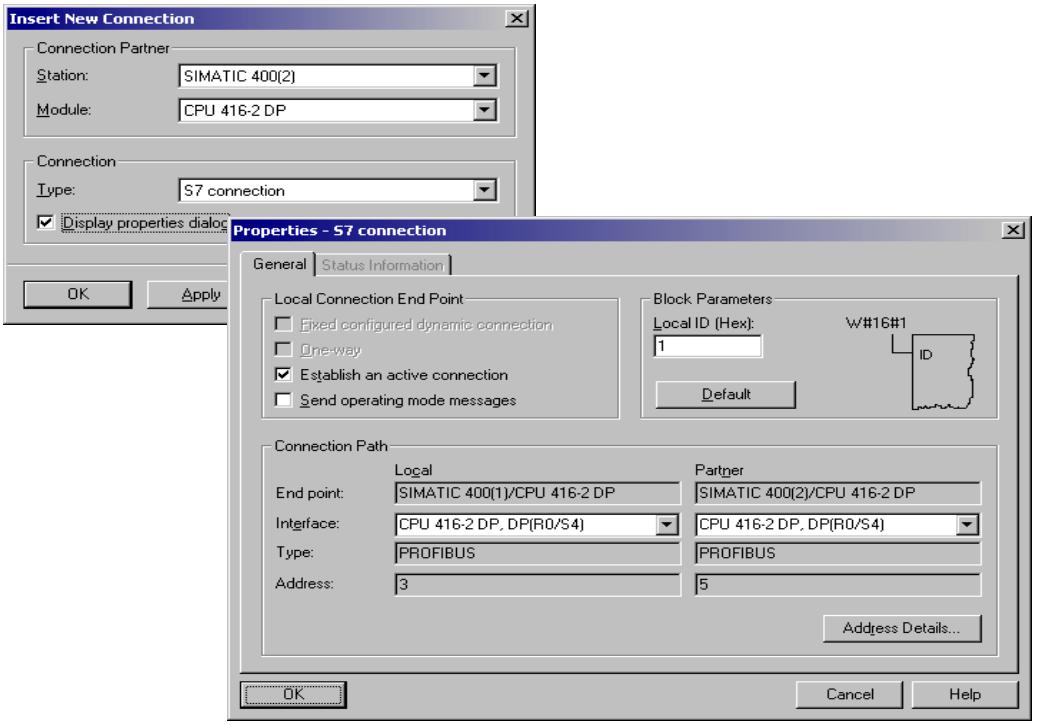


Рис. 2.6 Конфигурирование коммуникационных соединений

Если необходимо изменить дополнительные свойства соединения, активизируйте элемент управления check box "Show Properties Dialog" ("Показать диалоговое окно свойств").

Таблица соединений содержит все данные сконфигурированных соединений. Для точного отображения этих данных используйте опции меню: *View -> Display Columns (Вид -> Отобразить столбцы)*, после чего выберите интересующую Вас информацию.

Connection ID (идентификатор соединения)

Число устанавливаемых соединений определяется типом CPU. STEP 7 устанавливает ID для каждого соединения и для каждого партнера. Такая спецификация Вам потребуется при использовании коммуникационных блоков в Вашей программе.

Local ID (Локальный ID)

Вы можете модифицировать локальный ID (столбец **local ID** - ID соединения открытого в настоящий момент модуля). Такая необходимость может возникнуть, если Вы уже запрограммировали коммуникационные блоки и хотите использовать в них определенный локальный ID для соединения.

Вы должны ввести значение нового локального ID (local ID) в виде шестнадцатеричного числа. Оно может лежать внутри следующих диапазонов значений, в зависимости от типа соединения, и не должно быть к текущему моменту времени использовано:

- Диапазон значений для S7-соединений:

  0001_{16} … $0FFF_{16}$

- Диапазон значений для PtP-соединений:

  1000_{16} … 1400_{16}

Partner ID (ID партнера)

Вы можете также изменить ID партнера (столбец **partner ID**), перейдя к таблице соединений CPU партнера с последующим изменением локального ID (local ID); для этого необходимо выбрать строку с интересующим Вас соединением и затем использовать опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)*. Если STEP 7 не дает изменить ID партнера (partner ID), то это означает, что имеет место односторонняя связь (соединение) (см. ниже).

Partner (Партнер)

В данном столбце отображены коммуникационные партнеры. Если необходимо зарезервировать коммуникационный ресурс без указания имени устройства партнера, введите в окне ввода "Station" ("Станция") значение "unspecified" ("не определено").

При односторонней связи (**one-way connection**) инициализация коммуникаций производится со стороны только одного партнера; например: SFB-коммуникации между S7-400 и S7-300 CPU. Хотя коммуникационные SFB-функции не доступны для S7-300, в данном случае обмен данными может обеспечиваться S7-400 CPU с помощью SFB 14 GET и SFB 15 PUT. Для такого соединения не нужно выполнения программы пользователя в S7-300, так как управление обменом данными обеспечивается операционной системой.

Для односторонней связи (one-way connection) соединение конфигурируется с помощью таблицы соединений (connection table) "активного" ("active") CPU. Только после этого STEP 7 назначает локальный ID ("Local ID"). Вы должны загружать это соединение только в локальной станции.

При двусторонней связи (**two-way connection**) оба партнера могут проявлять коммуникационную активность; это могут быть, например, два S7-400 CPU. В данном случае обмен данными может обеспечиваться, например, с помощью SFB 8 BSEND и SFB 9 BRCV.

Для двусторонней связи (two-way connection) соединение конфигурируется только один раз для одного из двух партнеров. После этого STEP 7 назначает локальный ID ("Local ID"), ID партнера ("Partner ID") и генерирует коммуникационные данные для обеих станций. Вы должны загружать каждого партнера с его собственной таблицей соединений.

Type (Тип)

В данном столбце таблицы соединений устанавливается тип соединения.

Базовый пакет STEP 7 обеспечивает для конфигурирования сети следующие типы соединений (connection type):

Соединение **PtP connection** ("Point-to-point", "точка к точке") применяется для подсети PTP (процедуры 3964 (R) и RK 512) для SFB-коммуникаций. Соединение PtP служит для последовательной связи между двумя партнерами. Это могут быть два устройства SIMATIC S7 с соответствующими коммуникационными процессорами CP или одно устройство SIMATIC S7 и одно устройство стороннего производителя (не из семейства SIMATIC), например, принтер или считыватель штрих-кода.

Соединение **S7 connection** применяется для подсетей MPI, PROFIBUS и Industrial Ethernet с коммуникационными SFB-функциями. Соединение S7 обеспечивает связь между устройствами SIMATIC S7, включая программаторы PG и устройства HMI (устройства человеко-машинного интерфейса). Посредством соединения S7 производится обмен данными или выполняются функции управления или программирования.

Соединение **Fault-tolerant S7 connection** (отказоустойчивое соединение S7) применяется для подсетей PROFIBUS и Industrial Ethernet с коммуникационными SFB-функциями. Отказоустойчивое соединение S7 устанавливается между отказоустойчивыми устройствами SIMATIC S7 и может также устанавливаться для соответствующим образом оснащенного ПК.

Опционные пакеты "NCM S7 for PROFIBUS" и "NCM S7 for Industrial Ethernet" позволяют производить параметризацию коммуникационных процессоров CP.

В зависимости от установленного программного обеспечения NCM у Вас будут дополнительные типы соединений: FMS-соединение, FDL-соединение, ISO transport-соединение, TCP-соединение, ISO-on-TCP-соединение, UDP-соединение и E-mail-соединение.

Установка активного соединения (Active Connection Buildup)

Перед тем, как начать передачу данных, необходимо установить соединение (инициализировать). Если коммуникационные партнеры имеют такую возможность, то Вы можете задать устройство для установления соединения. Делается это с помощью элемента управления check box "Active connection buildup" ("Установка активного соединения") в окне свойств соединения: выделите соединение, затем выберите опции меню: *Edit -> Object Properties (Правка -> Свойства объекта*).

Посылка сообщений о рабочем состоянии (Sending operating state messages)

Коммуникационные партнеры с сконфигурированной двусторонней связью могут обмениваться сообщениями о рабочем состоянии. Если локальный узел должен посылать сообщения о своем рабочем состоянии, активируйте соответствующий элемент управления check box в окне свойств соединения. В программе пользователя CPU партнера эти сообщения могут приниматься с помощью SFB 23 USTATUS.

Расположение соединения (Connection Path)

Окно свойств соединения отображает конечные пункты соединения и подсети, через которые соединение осуществляется, в виде (адреса) расположения соединения (Connection Path). Если присутствует несколько подсетей на выбор, то STEP 7 выбирает их в следующей последовательности: сначала Industrial Ethernet, затем Industrial Ethernet/TCP-IP, затем MPI и, наконец, PROFIBUS.

Станция и CPU, через которые соединение осуществляется, отображаются как конечные пункты соединения. Модули с коммуникационными свойствами приводятся в списке в окне с пометкой "Interface" ("Интерфейс") с указанием номера стойки и номера слота. Если оба CPU расположены в одной стойке (например, два S7-400 CPU в многопроцессорном режиме), в окне отразится запись "PLC-internal" ("внутри PLC").

В окне с пометкой "Type" ("Тип") Вы можете выбирать подсети, через которые соединение должно осуществляться. Если оба коммуникационных партнера, например, подключены к одной MPI-подсети и к одной PROFIBUS-подсети, то в окне Вы увидите "MPI". Вы можете изменить эту спецификацию на "PROFIBUS", и STEP 7 автоматически примет остальные установки. После этого Вы увидите адрес MPI или адрес PROFIBUS для узла в окне "Address" ("Адрес").

Соединения между проектами (Connections between projects)

Для обмена данными между двумя S7 модулями, принадлежащими различным SIMATIC-проектам, Вы должны ввести значение "unspecified" ("не определено") для коммуникационного партнера в таблице соединений (в локальной станции в обоих проектах).

Убедитесь, что данные таблицы соединений согласованы, так как STEP 7 не проверяет согласованность этих данных самостоятельно. После сохранения и компиляции данных таблицы соединений Вы должны загрузить их в локальные станции в каждом проекте.

Соединения с не S7-станцией (Connection to non-S7 station)

В проекте Вы можете также определять станции, не относящиеся к S7-станциям, в качестве коммуникационных партнеров:

- Другие станции (устройства сторонних [не Siemens] производителей, а также S7-станции в других проектах)
- Программаторы PG / компьютеры (ПК)
- SIMATIC S5-станции

Необходимые условия для таких соединений заключаются в том, что другая (не S7) станция должна существовать как объект в каталоге проекта, кроме того должно быть выполнено подключение этой станции к соответствующей подсети в свойствах станции (например, выберите станцию с помощью утилиты конфигурирования сети Network Configuration, затем выберите опции: *Edit -> Object Properties* [*Правка -> Свойства объекта*] и подключите станцию к требуемой подсети на вкладке "Interfaces" ["Интерфейсы"]).

2.4.4 Переходы между подсетями (Network Transitions)

Если программатор подключен к подсети, он может иметь доступ ко всем узлам данной подсети. При этом из одной точки подключения Вы можете программировать и отлаживать программы для всех S7-станций, подключенных к MPI-сети. Если какая-либо S7-станция подключена также к другой подсети, такой как PROFIBUS, программатор может также иметь доступ ко всем узлам и этой подсети. Для этого должно выполняться требование, чтобы станция с переходом между подсетями имела возможность для программирования канала передачи фреймов сообщений.

Когда конфигурирование сети завершено, для станций с переходом между подсетями автоматически генерируются таблицы маршрутизации (routing table), содержащие всю необходимую информацию. Все доступные коммуникационные партнеры должны быть сконфигурированы в сети автоматизируемой установки в S7-проекте и должны "знать", к каким станциям имеется доступ и с помощью каких подсетей и переходов между подсетями.

Если необходимо для программатора, подключенного к подсети, обеспечить доступ ко всем узлам данной подсети из одной точки подключения, то Вы должны сконфигурировать соединение (точку подключения). Вы должны для этого ввести "placeholder" ("местодержатель"), PG/ПК станцию из каталога сетевых объектов (Network Object Catalog) в конфигурацию сети в соответствующей подсети. После этого PG/ПК станция должна быть сконфигурирована в каждой подсети, к которой необходимо будет подключать программатор PG.

Во время работы Вы будете подключать PG к подсети и выбирать опции меню: *PLC -> Assign PG/PC (PLC -> Назначить PG/ПК)*. Это позволяет настраивать интерфейсы программатора для работы с выбранной подсетью. Перед отключением PG от подсети требуется выбрать опции меню: *PLC -> Undo PG/PC Assignment (PLC -> Отменить назначение PG/ПК)*.

Если необходимо перейти в интерактивный (online) режим с программатором, в котором нет необходимого проекта, Вам потребуется ID S7-подсети для доступа к сети. ID S7-подсети содержит два номера: номер проекта и номер подсети. Вы можете получить ID подсети из данных конфигурации сети. Для этого выберите сначала подсеть, затем - опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)* на вкладке "General" ("Общие").

2.4.5 Загрузка таблицы соединений (Loading the Connection Data)

Для активации соединений Вы должны сохранить, скомпилировать и загрузить таблицу соединений ("connection table") в PLC (все таблицы соединений во все "активные" CPU).

Требование: Когда Вы открываете окно сети (network window), таблица соединений отражается на экране. Программатор является узлом подсети, с помощью которой могут быть загружены данные таблицы соединений в модули с коммуникационными свойствами. Все узлы подсети имеют уникальные адреса. Все модули, в которые должны быть переданы данные таблицы соединений, должны находиться в режиме STOP.

С помощью опций меню: *PLC -> Download -> … (PLC -> Загрузить -> …)* Вы можете передать данные таблицы соединений и данные конфигурации в доступные модули. В зависимости от того, какой объект и какие команды мены выбраны, Вы можете выбирать из следующего ряда опций:

-> *Selected Stations (Выбранные станции)*

-> *Selected and Partner Stations (Выбранные и станции партнера)*

-> *Selected Connections (Выбранные соединения)*

-> *Stations on Subnet (Станции в подсети)*

-> *Connections and Gateways (Соединения и шлюзы)*

Для удаления всех данных таблицы соединений в программируемом модуле загрузите в него пустую таблицу соединений (connection table).

Скомпилированные данные таблицы соединений также являются частью системных данных (*System data*) в каталоге *Blocks*. Передача системных данных и последующий запуск CPU приводит к передаче данных таблицы соединений в модули с коммуникационными свойствами.

Для интерактивной (online) работы с помощью MPI для программатора не требуется дополнительного оборудования. Если же к сети Вы подключаете ПК или подключаете PG к сетям Ethernet или PROFIBUS, то Вам потребуются соответствующие интерфейсные модули. Параметризация модулей производится с помощью приложения "Set PG/PC Interface" из панели управления Windows.

2.5 Создание S7-программ

2.5.1 Введение

Программа пользователя создается в каталоге (в объекте) *S7 Program.* Вы можете назначать этот объект в объекте CPU в структурной иерархии проекта, или вне зависимости от CPU. В свою очередь объект *S7 Program* включает в себя объект *Symbols (Символы)* и каталоги *Source Files (Исходные файлы)* и *Blocks (Блоки)* (см. рис. 2.7).

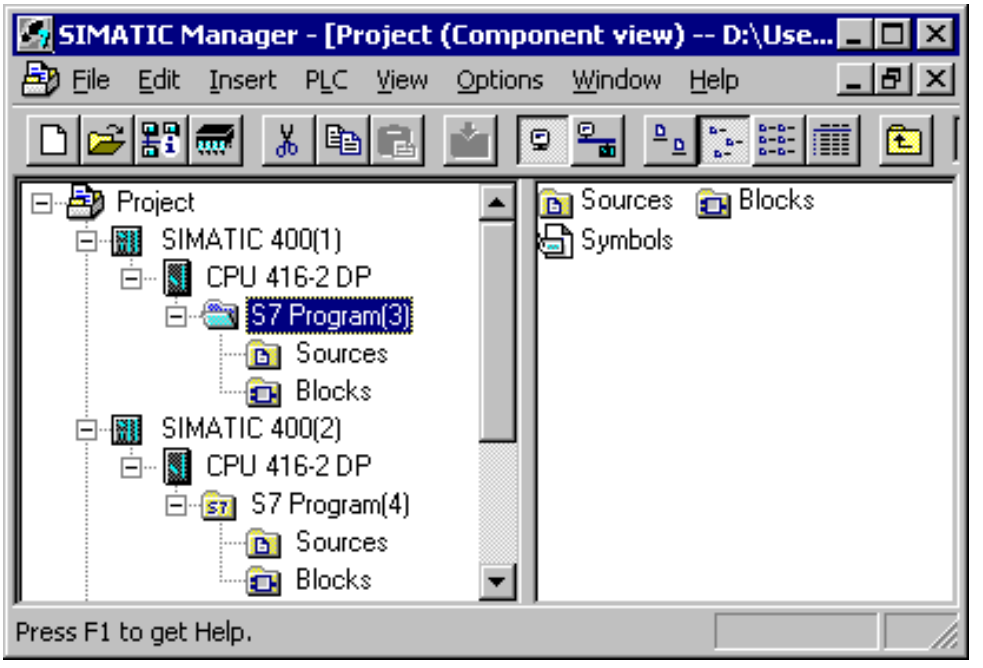


Рис. 2.7 Объекты, участвующие при генерации программы.

В случае создания программы путем написания исходных файлов ("**source-oriented**") Вы должны создать одну или несколько исходных программ и сохранить их в виде файлов в каталоге *Source Files (Исходные файлы)*. Исходные программы - это текстовые файлы формата ASCII, которые содержат операторы программы для одного или нескольких блоков, возможно даже целиком всю программу. Вы должны скомпилировать исходные программы; скомпилированные блоки программы помещаются в каталог *Blocks (Блоки).* Скомпилированные блоки содержат код MC7 и выполняются в S7 CPU.

В случае создания программы "инкрементным" путем ("**incremental**"), - методом добавления - Вы вводите программу блок за блоком. Вводимые блоки немедленно проверяются на наличие синтаксических ошибок. При поступлении команды на сохранение блок сначала компилируется, затем сохраняется в каталоге *Blocks (Блоки).* При создании программы данным методом Вы можете также редактировать блоки в интерактивном (online) режиме в CPU, даже во время рабочего режима.

В программе обрабатываются значения сигналов или значения адресов. Адрес - это, например, вход I1.0 (абсолютная адресация). С помощью таблицы символов **Symbol Table** в объекте Symbols, Вы можете назначить адресу символьное имя, например, "Switch motor on" ("Включение мотора") и после этого обращаться к этому адресу, используя данное символьное имя (символьная адресация). В свойствах автономного объекта *Blocks (Блоки)* Вы можете определить, каким способом будут адресоваться переменные в таблице символов (Symbol Table) после корректировки - абсолютным или символьным в уже скомпилированных блоках, согласно приоритету адресации (*address priority*).

Требования к памяти

Требования к памяти для скомпилированного блока можно найти в свойствах блока для этого при выбранном в SIMATIC Manager блоке, выберите опции меню: *Edit -> Object Properties (Правка -> Свойства объекта),* вкладка "General-Part 2" ("Общие - часть 2").

Вы можете узнать требования к памяти для Вашей программы в целом, если выбрать в SIMATIC Manager программу из объекта *Blocks (Блоки)* и затем выбрать опции меню: *Edit -> Object Properties (Правка -> Свойства объекта).* На вкладке *Blocks (Блоки)* Вы найдете размер программы в загрузочной памяти (load memory) и в рабочей памяти (work memory), так же как и число блоков каждого типа.

Системные данные не рассматриваются среди этой информации; они занимают дополнительное пространство в загрузочной памяти.

2.5.2 Таблица символов (Symbol Table)

В управляющей программе Вы имеете дело с адресами, т.е. с входами, с выходами, таймерами и блоками. Вы можете назначать абсолютные адреса (например, I1.0) или символьные адреса (например, Start signal [сигнал запуска]). При символьной адресации используются символьные имена. Это делает программу легко читаемой, благодаря тому, что символьные имена несут смысловую нагрузку.

При использовании символьной адресации различаются локальные *(local)* и глобальные *(global)* символы (символьные имена). Локальный *(local)* символ распознается только в блоке, в котором они определены. Поэтому при необходимости Вы можете использовать одинаковые локальные символьные имена в различных целях в разных блоках. Глобальный символ распознается в любом месте программы и имеет одинаковое значение во всех блоках программы. Вы должны определить глобальный символ в таблице символов (объект *Symbols* в каталоге *S7 Program*).

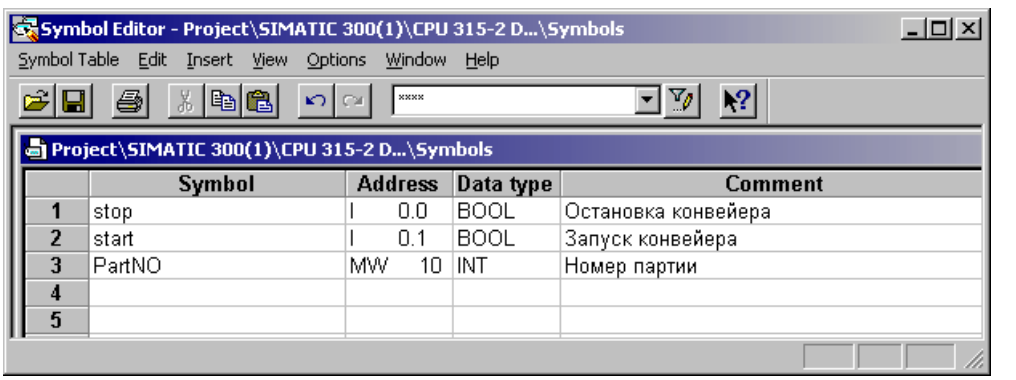
Глобальный символ начинается с символа алфавита и может иметь в длину до 24 символов. Глобальный символ может также содержать пробелы, специальные символы и национальные символы, например, такие как умляут. Исключения составляют символы 00 $_{hex}$, FF $_{hex}$ и кавычки ("). При программировании Вы должны заключать спецсимволы в кавычки. В скомпилированном блоке программный редактор отображает все глобальные символы в кавычках. Комментарий к символу может составлять в свою очередь запись из 80 символов.

В таблице символов Вы можете назначать имена следующим адресам и объектам:

- Входам I, выходам Q, периферийным входам PI и выходам PQ
- Меркерам M, таймерам T и счетчикам C
- Блокам кодов OB, FB, FC, SFC, SFB и блокам данных DB
- Типам данных, определенным пользователем, UDT
- Таблице переменных VAT

Адреса данных в блоках данных находятся среди локальных адресов; связанные символы определяются в разделе описаний (declaration section) блоков данных в случае глобальных блоков данных и в разделе описаний (declaration section) функциональных блоков в случае экземплярных блоков данных.

При создании S7-программ SIMATIC Manager создает также пустую таблицу символов *Symbols*. Вы можете открыть эту таблицу и определить глобальные символы и назначить их абсолютным адресам (рис. 2.8).



	Symbol	Address	Data type	Comment
1	stop	I 0.0	BOOL	Остановка конвейера
2	start	I 0.1	BOOL	Запуск конвейера
3	PartNO	MW 10	INT	Номер партии
4				
5				

Рис. 2.8 Пример таблицы символов Symbol Table

В S7-программе может быть только одна таблица символов *Symbols.*

Тип данных является частью определения символа. Он определяет особые свойства данных, в частности представление содержимого данных. Например, тип данных BOOL идентифицирует двоичную переменную, а тип данных INT обозначает переменную в цифровой форме, содержание которой определяется 16-битным целым числом. Для получения более подробной информации обратитесь к разделу 3.7 "Переменные и константы" и к разделу 24 "Типы данных", содержащим соответственно обзор и подробное описание типов данных в STEP 7.

В случае "инкрементного" программирования Вы создаете таблицу символов до ввода программы; Вы можете также добавить или скорректировать отдельные символы во время ввода программы. При создании программы путем, ориентированным на создание исходных текстов программы готовая таблица символов должна быть доступна к моменту компиляции программы.

Импорт, экспорт

Таблица символов может быть импортирована и экспортирована. Здесь "экспортируемый" файл означает созданный файл, содержащий данные Вашей таблицы символов. Здесь может быть как таблица символов целиком, так и отдельные строки таблицы. Для файла Вы можете выбирать следующие форматы данных: текстовый ASCII (с расширением *.asc), sequential assignment list (последовательный список выражений - с расширением *.seq), System Data Format (формат системных данных - с расширением *.sdf для Microsoft Access) и Data Interchange Format (формат обмена данными - с расширением *.dif для Microsoft Excel). Вы можете редактировать экспортируемый файл с помощью подходящего редактора. Вы можете также импортировать таблицу символов в одном из выше упомянутых форматов.

Специальные свойства объектов

Выбрав опции меню: *Edit -> Special Object Properties (Правка -> Специальные свойства объекта),* Вы можете установить атрибуты для каждого символа в таблице символов. Эти атрибуты или свойства используются для следующих целей:

- Для HMI функций для мониторинга с использованием WinCC
- Для конфигурирования коммуникаций
- Для конфигурирования сообщений
- Для мониторинга процесса посредством S7-PDIAG

Выбрав опции меню: *View -> Columns O, M, C, R (Вид -> Столбцы O, M, C, R),* Вы сделаете атрибуты видимыми. С помощью опций меню: *Options -> Customize (Опции -> Установки пользователя)* Вы можете установить, будут или нет специальные свойства объектов копироваться, и Вы сможете определить поведение при импортировании символов.

2.5.3 Редактор STL-программ (STL Program Editor)

Для создания программы пользователя в базовый пакет STEP 7 (STEP 7 Basic Package) включены редакторы для языков программирования LAD, FBD и STL. При работе с редактором STL-программы Вы можете вводить программу "инкрементно" (непосредственно) или генерировать исходный

текст программы и компилировать его позднее. На рис. 2.9 показаны возможные действия, связанные с созданием STL-программы.

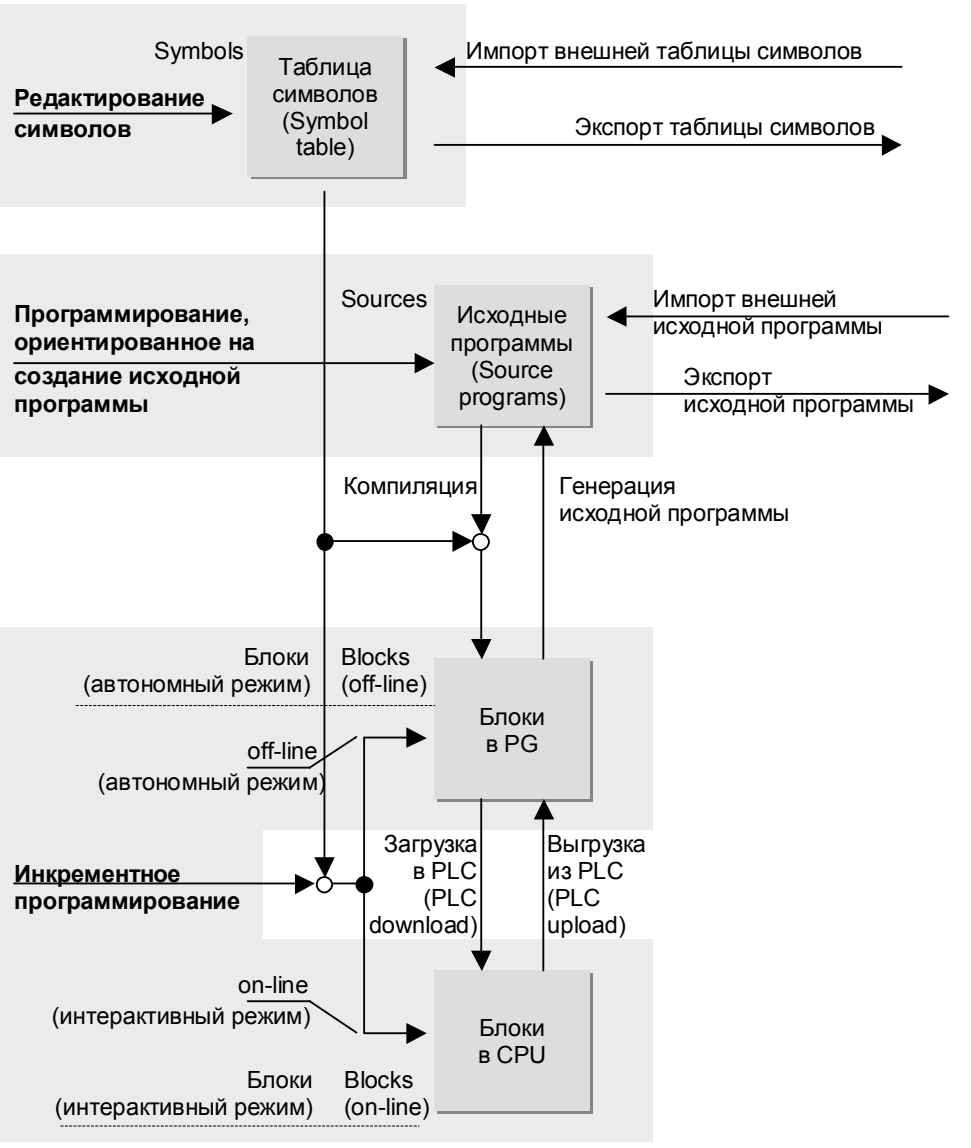


Рис. 2.9 Создание программы с использованием редактора STL Editor.

Если при инкрементном программировании используется символическая адресация глобальных адресов, то предварительно должно быть выполнено присвоение символических имен абсолютным адресам. Тем не менее, Вы можете вводить новые символы или изменять ранее присвоенные во время ввода программы. В случае программирования, ориентированного на создание исходной программы, к моменту компиляции программы таблица символов должна быть заполнена.

STL-блоки могут быть "декомпилированы" ("decompiled"), т.е. из MC7 кода может быть извлечен пригодный для чтения блок без автономной базы данных (offline database) (Вы можете прочитать любой блок из CPU, используя программатор PG без связанного проекта). Вдобавок, исходная STL- программа может быть восстановлена из любого скомпилированного блока.

Запуск редактора STL- программ

Вы можете получить доступ к редактору программ при открытии блока в SIMATIC Manager, например, двойным щелчком на автоматически сгенерированном символе для организационного блока OB1 или с помощью меню панели задач Windows: *Start -> Simatic -> STEP 7 -> LAD, STL, FBD - Program S7 Blocks.*

Вы можете задать свойства для редактора программ с помощью опций меню: *Options -> Customize (Опции -> Установки пользователя).* На вкладке "Editor" ("Редактор") выберите свойства, с которыми новый блок должен быть сгенерирован и отображен, такие, например, как язык создания, установки для комментариев и символов.

При открытии скомпилированного блока в каталоге *Blocks (Блоки) (например, двойным щелчком)* блок открывается для инкрементного программирования. Для программирования, ориентированного на создание исходных текстов программы, Вы должны открывать исходный файл программы в каталоге *Source files (Исходные файлы).*

Вы можете также создавать программу, используя попеременно то один метод, то другой метод программирования, т.е. некоторые блоки вводятся непосредственно, а другие блоки создаются с помощью исходных файлов. Также в программе можно вызывать отдельные блоки, созданные с использованием других языков программирования, таких как LAD и FBD. Программа пользователя создается блок за блоком и в результате представляет собой исполняемую программу в коде MC7 независимо от языка программирования.

Для создания программы пользователя рекомендуется применять метод, ориентированный на создание исходных текстов программы, с использованием символьной адресации. Редактирование получается проще, меньше случается синтаксических ошибок и можно использовать какой-либо другой текстовый редактор. С помощью таблицы символов Вы можете определять различные абсолютные адреса всякий раз перед компиляцией программы, так, что Вы можете создавать многократно используемые "стандартные программы" независимо от конфигурации оборудования.

Способ программирования, ориентированный на создание исходных текстов программы, является единственно возможным способом, обеспечивающим блокам Вашей программы защиту (block protection KNOW_HOW_PROTECT).

Инкрементное программирование, тем не менее, является оптимальным для быстрой проверки изменений в программе непосредственно в CPU. Если изменение программы выдержало проверку, обновите и вновь скомпилируйте исходную программу. Таким образом, у Вас всегда будет текущая версия программы в формате ACSII-текстового файла. Инкрементное программирование также очень удобно для тестирования программы с помощью нескольких операторов, включаемых в интерактивном (online) режиме, которые в дальнейшем (после отладки) использоваться не будут.

Способ, ориентированный на создание исходных текстов программы

Способ программирования "Source-oriented" (ориентированный на создание исходных текстов программы) используется для редактирования исходных файлов STL-программы в каталоге *Source Files (Исходные файлы).* STL-файл имеет формат чисто ACSII-текстового файла. Он может содержать исходную программу для одного или нескольких блоков данных или блоков кода, также как определения данных пользовательского типа.

В SIMATIC Manager выберите каталог *Source Files (Исходные файлы)* и создайте новый исходный файл с помощью опций меню: *Insert -> S7 Software -> STL Source File (Вставка -> ПО S7 -> Исходный STL-файл).*

Можно создавать новые блоки намного более простым способом, если использовать опции меню: *Insert -> Block Template -> … (Вставка -> Шаблон блока -> …)* (в редакторе). Программа-редактор использует шаблоны из каталога …\Step7\S7ska, которые находятся в текстовых файлах S7kaf*nn*x.txt. Вы можете привести эти шаблоны к виду, отвечающему Вашим требованиям.

Вы также можете выбрать в качестве способа создания нового исходного STL-файла из одного или нескольких скомпилированных блоков с помощью опций меню: *File -> Generate Source File (Файл -> Создать исходный файл).*

Если Вы создали исходный файл с помощью стороннего текстового редактора, Вы можете использовать опции из меню SIMATIC Manager: *Insert -> External Source File (Вставка -> Внешний исходный файл)* для помещения файла в каталог *Source Files (Исходные файлы).* Вы можете скопировать выбранный исходный файл в каталог по Вашему выбору с помощью опций: *Edit -> Export Source File (Правка -> Экспорт исходного файла).*

При программировании, ориентированном на создание исходных файлов программы, Вы должны соблюдать определенные правила и использовать ключевые слова, зарезервированные для данного компилятора. В разделах 3.4.3 "Программирование кодовых блоков, ориентированное на создание исходных файлов на STL" и 3.6.2 "Программирование блоков данных, ориентированное на создание исходных файлов" представлена структура исходного STL-файла.

Компилирование исходного STL-файла

Вы можете сохранить исходную программу в любой момент во время редактирования, даже если программа еще не закончена. Программный редактор генерирует исполняемые блоки только по завершении создания программы и помещает их в каталог *Blocks (Блоки).* Если необходимо использовать глобальные символы в исходном STL-файле, то скомпилированная таблица символов должна быть доступна к моменту его компиляции.

На вкладке "Source Files" ("Исходные файлы") в диалоге, вызванном с помощью опций меню: *Options -> Customize (Опции -> Установки пользователя)* Вы можете задать свойства компилятора, такие как выбор условия компиляции исходных блоков, определяющего, должны ли переписываться существующие блоки, или блоки должны обновляться только, когда вся программа целиком будет свободна от ошибок. На вкладке "Generate Block" ("Создать блок") Вы можете задать автоматическое обновление ссылок при компиляции блока.

С помощью опций меню: *File -> Consistency Check (Файл -> Проверка соответствия)* можно осуществить синтаксическую проверку программы без

компилирования блоков.

Если исходная программа открыта, Вы можете начать компилирование посредством опций меню: *File -> Compile (Файл -> Компиляция).* Все не содержащие ошибок блоки исходной программы будут скомпилированы. Также как любой блок, содержащий ошибки не будет скомпилирован. Если при компиляции было выдано предупреждение, блок будет скомпилирован в любом случае, но, тем не менее, выполнение программы в CPU, возможно, будет с ошибками.

Вызываемые блоки должны уже присутствовать в скомпилированном виде при компиляции программы, или они должны располагаться в программе до точки их вызова (для более детального освещения вопроса о порядке расположения блоков обратитесь к разделу 3.4.3 "Программирование кодовых блоков, ориентированное на создание исходных файлов на STL").

Обновление или генерирование исходного STL-файла

На вкладке "Source Files" ("Исходные файлы") в диалоге, вызванном с помощью опций меню: *Options -> Customize (Опции -> Установки пользователя)* Вы можете выбрать опцию "Generate source files automatically" ("Создать исходный файл автоматически"), так что при сохранении запрограммированного инкрементным способом блока будет обновлен ранее существовавший исходный файл программы или будет сгенерирован новый файл (если таковой ранее не существовал). Вы можете задать имя нового исходного файла программы исходя из абсолютного или символьного адреса. Исходному файлу программы могут быть переданы адреса в абсолютной или символьной форме.

С помощью кнопки "Execute" ("Выполнение") Вы выбираете в последующем диалоговом окне блок, с которого Вы желаете начать генерацию исходного файла программы.

Инкрементное программирование

При инкрементном программировании Вы можете редактировать блоки и в автономном (offline), и в интерактивном (online) каталоге *Blocks (Блоки)*. При данном методе программирования редактор проверяет введенные в программу изменения сразу же после завершения текущей строки программы. При закрытии блока, он немедленно компилируется, так что сохранить можно только блоки, не содержащие ошибок.

На вкладке "Create Block" ("Создать блок"), выбранной с помощью опций меню: *Options -> Customize (Опции -> Установки пользователя)*, Вы можете выбрать опцию автоматического обновления ссылок (reference data) при создании блока.

Блоки могут редактироваться в автономном (offline) режиме в базе данных программатора и в интерактивном (online) режиме в CPU (или, как обычно говорится, в программируемом контроллере ["programmable controller"] или в PLC). Для этих целей в SIMATIC Manager используются два окна ("offline" и "online"); окна отличаются друг от друга строкой заголовка.

В "автономном" ("offline") окне Вы можете редактировать блоки непосредственно в базе данных PG. Из среды редактора Вы можете сохранить измененный блок в автономной ("offline") базе данных с помощью опций меню: *File -> Save (Файл -> Сохранить)* и передать его в CPU посредством опций меню: *PLC -> Download (PLC -> Загрузить).* Если нужно

сохранить открытый блок с другим номером или в другом проекте, если необходимо переслать его в библиотеку или в другой CPU, то Вы должны использовать команду: *File -> Save as (Файл -> Сохранить как).*

Для редактирования блока в CPU откройте блок в "интерактивном" ("online") окне. Это действие перешлет блок из CPU в программатор для редактирования. Вы можете вновь переслать отредактированный блок в CPU командой: *PLC -> Download (PLC -> Загрузить).* Если CPU находится при этом в рабочем (RUN) режиме, то обработка отредактированного блока начнется со следующего цикла сканирования программы. Если необходимо сохранить блок, отредактированный в "интерактивном" ("online") режиме также и в автономной ("offline") базе данных, то это можно сделать с помощью опций меню: *File -> Save (Файл -> Сохранить).*

В разделах 2.6.4 "Загрузка программы пользователя в CPU" и 2.6.5 "Обработка блока" содержится дополнительная информация по интерактивному (online) программированию. В разделах 3.4.2 "Инкрементное программирование кодовых блоков на STL" и 3.6.1 "Инкрементное программирование блоков данных на STL" показано, как вводится STL-блок.

2.5.4 Редактор SCL-программ (SCL Program Editor)

Опционное программное обеспечение S7-SCL дает возможность пользователю создавать программы на языке программирования SCL. При инсталляции ПО S7-SCL этот редактор встраивается в SIMATIC Manager. Вы можете использовать его точно также как и редакторы для стандартных языков программирования. Работая с SCL, Вы будете использовать метод программирования, ориентированный на создание исходных файлов программы (см. рис. 2.10).

Вы создаете сначала исходный текст программы, который затем необходимо скомпилировать. Вы можете также вызывать ранее скомпилированные блоки, находящиеся в каталоге *Blocks (Блоки),* встраивая их в Вашу программу. Эти блоки могут быть написаны с использованием иного языка программирования, например, на STL.

Если в программе Вы используете символьную адресацию для глобальных адресов, то готовая таблица символов уже должны быть доступна к моменту компиляции программы.

Однако Вы не сможете сгенерировать исходный SCL-файл из скомпилированного блока, например, если Вы удалили исходный файл по ошибке. (Примечание: скомпилированный блок будет выполняться в CPU, даже если исходный файл программы блока станет недоступным).

Запуск редактора STL- программ

Редактор SCL-программ запускается при открытии в SIMATIC Manager скомпилированного SCL-блока или исходного SCL-файла или с помощью меню панели задач Windows: *Start -> Simatic -> STEP 7 -> S7-SCL -> Program S7 Blocks.*

Если программный редактор при открытии скомпилированного SCL-блока не находит соответствующего ему исходного файла программы, например, если этот исходный файл был удален или перемещен, то блок будет открыт для редактирования с помощью редактора STL-программ.

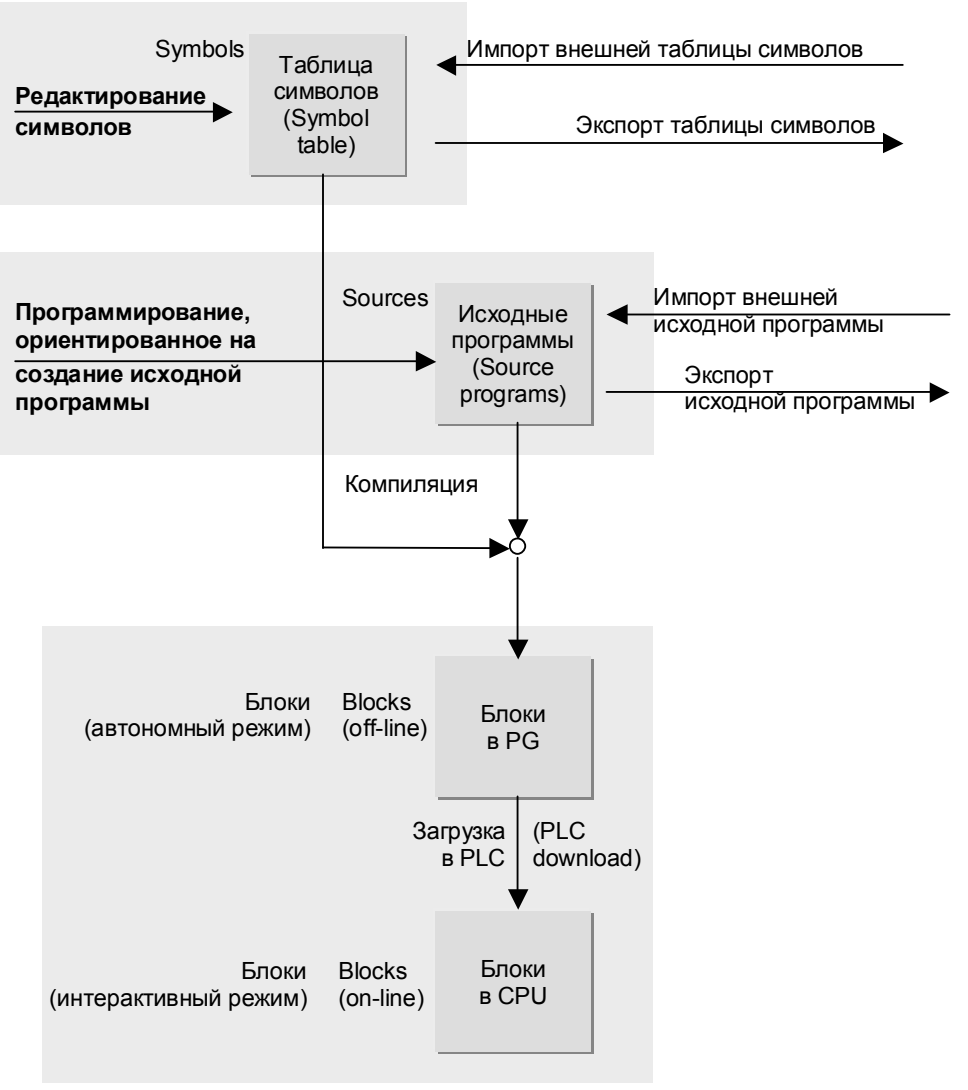


Рис. 2.10 Создание программы с использованием редактора SCL Program Editor.

Вы можете задать свойства для редактора SCL-программ с помощью опций меню: *Options -> Customize (Опции -> Установки пользователя).* На вкладке "Editor" ("Редактор") выберите свойства, с которыми новый блок должен быть сгенерирован и отображен, такие, например, как отображение программы с нумерацией строк.

Создание исходного SCL-файла

Выбрав каталог Source files (Исходные файлы) в SIMATIC Manager и затем опции меню: *Insert -> S7 Software -> SCL Source File (Вставка -> ПО S7 -> Исходный SCL-файл),* можно создать новый исходный файл. Двойным щелчком на исходном файле Вы можете его открыть. Простой способ создания нового блока возможен также при использовании опций меню: *Insert -> Block Template -> … (Вставка -> Шаблон блока -> …),* а также Вы можете

вставить готовые программные структуры в исходный программный файл в позиции курсора.

Если Вы создали исходный SCL-файл с помощью стороннего текстового редактора, Вы можете использовать опции из меню SIMATIC Manager: *Insert -> External Source File (Вставка -> Внешний исходный файл)* для помещения файла в каталог *Source Files (Исходные файлы).* Вы можете скопировать выбранный исходный файл в каталог по Вашему выбору с помощью опций: *Edit -> Export Source File (Правка -> Экспорт исходного файла).*

Если измененный исходный файл еще не сохранен, это индицируется символом звездочки после имени этого файла на панели заголовка окна редактора или в меню "Window" ("Окно").

При программировании, ориентированном на создание исходных файлов программы, Вы должны соблюдать определенные правила и использовать ключевые слова, зарезервированные для данного компилятора. В разделах 3.5.2 "Программирование кодовых блоков на SCL" и 3.6.2 "Программирование блоков данных, ориентированное на создание исходных файлов" представлена структура исходного SCL-файла.

Компилирование исходного SCL-файла

Вы можете сохранить исходную программу в любой момент во время редактирования, даже если программа еще не закончена. Программный редактор генерирует исполняемые блоки только по завершении создания программы и помещает их в каталог *Blocks (Блоки)*. Если необходимо использовать глобальные символы в исходном SCL-файле, то скомпилированная таблица символов должна быть доступна к моменту его компиляции.

Вы можете задать следующие установки среди других на вкладке "Compiler" ("Компилятор") в диалоговом окне, вызванном с помощью опций меню: *Options -> Customize (Опции -> Установки пользователя).*

- Create object code (Создать объектный код):
  Если выбрана данная опция, генерируются программные блоки, при условии, что не было выявлено ошибок при компиляции; с другой стороны Вы можете проверить программу на наличие синтаксических ошибок без создания блоков.
- Optimize object code (Оптимизировать объектный код):
  Если выбрана данная опция, созданные программные блоки оптимизируются с учетом требований к памяти и времени их выполнения.
- Monitor array limits (Отслеживать размерность массивов):
  Если выбрана данная опция, компилятор генерирует дополнительный программный код, который позволяет проверять, например, размерность массивов во время выполнения программы.
- Create debug info (Создать информацию для отладки):
  Если все еще необходимо отлаживать скомпилированную программу с помощью Program Status, Вы выбираете эту опцию. (Эта информация создается внутренне - без сохранения дополнительных программ)
- Set OK flag (Установить бит OK):
  Вы должны выбрать данную опцию, если в программе используется переменная OK или механизм EN/ENO.

Если исходный файл программы открыт, Вы можете начать компилирование посредством опций меню: *File -> Compile (Файл -> Компиляция).* Все не содержащие ошибок блоки исходной программы будут скомпилированы. Также как любой блок, содержащий ошибки не будет скомпилирован. Если при компиляции было выдано предупреждение, блок будет скомпилирован в любом случае, но, тем не менее, выполнение программы в CPU, возможно, будет с ошибками. Если необходимо скомпилировать только отдельные избранные блоки, выберите опции: *File -> Partial Compile (Файл -> Частичная компиляция).*

Вызываемые блоки должны уже присутствовать в скомпилированном виде при компиляции программы, или они должны располагаться в программе до точки их вызова (для более детального освещения вопроса о порядке расположения блоков обратитесь к разделу 3.5.2 "Программирование кодовых блоков на SCL"). SCL-компилятор автоматически создает отсутствующие экземпляры DB, если вызываются функциональные блоки. Для DB номер берется из таблицы символов (Symbol Table) или же выбирается наименьший свободный номер.

При компилировании стандартные блоки, такие, например, как IEC-функции, при первом вызове копируются в каталог *Blocks (Блоки)* из стандартной библиотеки.

С помощью опций: *PLC -> Download (PLC -> Загрузить)* можно загрузить в подключенный CPU все блоки, которые были созданы или были скопированы из стандартной библиотеки в каталог *Blocks (Блоки)* и скомпилированы при последней компиляции программы.

Компилирование файла управления

Язык программирования SCL облегчает процесс компиляции нескольких исходных файлов, которые должны выполняться вместе, но в определенном порядке. Вы должны создать файл управления компиляцией с помощью выбора опций меню: *Insert -> SCL Compilation Control File (Вставка -> Файл управления компиляцией)* при выбранном каталоге *Source Files (Исходные файлы).*

Откройте файл управления компиляцией и определите с помощью названий исходных файлов порядок, в котором эти файлы должны быть скомпилированы.

Посредством опций меню: *File -> Compile (Файл -> Компиляция)* Вы можете начать процедуру компиляции.

2.5.5 Перекомпоновка (Rewiring)

Функция перекомпоновки *Rewiring* позволяет Вам изменить адреса в отдельно скомпилированных блоках или в пользовательской программе в целом. Например, Вы можете изменить входные биты I 0.0 … I 0.7 на входные биты I 16.0 … I 16.7. Допустимыми адресами являются входы, выходы, меркеры, таймеры и счетчики, а также функции FC и функциональные блоки FB.

В SIMATIC Manager Вы должны выбрать объекты, в которых необходимо выполнить перекомпоновку; выберите отдельный блок, группу блоков, удерживая клавишу Ctrl, и щелкните кнопкой мыши на этих объектах или на пользовательской программе в целом - на каталоге *Blocks (Блоки).*

Для работы с таблицей, в которой необходимо указать старые адреса и адреса замены, выберите опции меню: *Options -> Rewire (Опции -> Перекомпоновка).* После того, как Вы подтвердите щелчком по кнопке OK, сделанные изменения, SIMATIC Manager выполнит замену адресов. После этого информационный файл покажет, в каких блоках были сделаны изменения и в каком количестве.

Существуют также дополнительные способы перекомпоновки:

- Для отдельно скомпилированных блоков Вы можете использовать также функцию *Address priority (Приоритет адреса).*
- В случае использования метода программирования, ориентированного на создание исходных текстов программы с использованием символьной адресации, Вы можете до компиляции внести изменения в таблицу символов, и после компиляции Вы получите перекомпонованную (*rewire*) программу.

2.5.6 Приоритет адресов (Address Priority)

В окне свойств автономного ("offline") объекта *Blocks (Блоки)* на вкладке "Blocks" ("Блоки") Вы можете назначить приоритет для одного из типов адресации (абсолютной или символьной) для уже сохраненных блоков, если эти блоки отображены. Перед этим все изменения в таблице символов или все назначения блоков глобальных данных должны быть сохранены.

По умолчанию принята установка: "Absolute address has priority" (абсолютная адресация имеет приоритет) (такое же поведение, как и в предыдущих версиях STEP 7). Такая установка означает, что если делаются изменения в таблице символов, то остается в силе абсолютная адресация, а символьные адреса соответственно изменяются.

Если задана установка: "Symbol has priority" (символьная адресация имеет приоритет), то это означает, что остается в силе символьная адресация, а абсолютные адреса изменяются.

Пример:

Пусть таблица символов содержит следующую информацию:

I 1.0 "Limit_switch_up" ("Верхний концевой переключатель")

I 1.1 "Limit_switch_down" ("Нижний концевой переключатель"),

и в программе скомпилированного блока сканируется вход I 1.0:

A **I 1.0 "Limit_switch_up"**

Если задана установка: "Symbol has priority" (символьная адресация имеет приоритет), а в таблице символов поменять значения выше указанных входов следующим образом:

I 1.0 "Limit_switch_down" ("Нижний концевой переключатель")

I 1.1 "Limit_switch_up" ("Верхний концевой переключатель"),

то программа будет содержать строку:

A I 1.1 **"Limit_switch_up"** ("Верхний концевой переключатель"),

а если задана установка: "Absolute address has priority" (абсолютная адресация имеет приоритет), то программа будет содержать строку:

A **I 1.0** "Limit_switch_down" ("Нижний концевой переключатель")

Если в результате изменений в таблице символов нет больше никаких назначений символьных имен абсолютным адресам, тогда при установке "Absolute address has priority" (абсолютная адресация имеет приоритет), выражения в программе будут содержать абсолютную адресацию (даже в режиме отображения символов), так как символьные имена как таковые отсутствуют. Если при тех же условиях задана установка: "Symbol has priority" (символьная адресация имеет приоритет), тогда те же выражения будут отброшены как ошибочные (так как обязательный абсолютный адрес игнорируется).

Если задана установка: "Symbol has priority" (символьная адресация имеет приоритет), то инкрементно программируемые блоки с символической адресацией сохранят свои символы в случае изменений в таблице символов. В этом случае такой блок с уже готовой программой может быть перекомпонован (rewired) с помощью изменения назначения адресов.

Примечание: такая перекомпоновка (rewiring) не может выполняться автоматически, так как скомпилированные блоки содержат исполняемый MC7-код с абсолютной адресацией. Изменения делаются только в соответствующих блоках при поступлении соответствующего сообщения, после которого они могут быть открыты и окончательно вновь сохранены.

2.5.7 Ссылки (Reference Data)

В добавление к собственно программе утилита SIMATIC Manager предоставляет Вам ссылки (Reference Data), которые Вы можете использовать как основу для корректировки или тестирования программы. Ссылочные данные содержат:

- Cross references (Перекрестные ссылки)
- Reserved locations (Зарезервированные области: I, Q, M, T, C)
- Program structure (Структура программы)
- Unused symbols (Неиспользуемые символы)
- Addresses without symbols (Адреса без символов)

Для генерации ссылочных данных выберите объект *Blocks (Блоки)* и затем опции меню: *Options -> Reference Data -> Display (Опции -> Ссылочные данные -> Отобразить)*. Представление ссылочных данных может изменяться в соответствии с рабочим окном с помощью опций: *View -> Filter… (Вид -> Фильтр…);* Вы можете сохранять установки для последующих сессий редактирования с помощью опций: *Save as Standard (Сохранить как стандарт).* Вы можете выводить на экран одновременно несколько списков.

С помощью опций меню: *Options -> Customize (Опции -> Установки пользователя)* в редакторе программы на вкладке "Create Blocks" ("Создать блоки") Вы можете включить или отменить обновление ссылочных данных при компиляции исходного файла программы или при сохранении инкрементно введенного блока.

Примечание: ссылки доступны только при автономной (offline) обработке данных; "автономные" (offline) ссылочные данные могут отображаться даже, если функция вызывается в блоке, открытом интерактивно (online).

Перекрестные ссылки (Cross references)

Список перекрестных ссылок показывает использование адресов и блоков в программе пользователя. В него входят абсолютные адреса, символьные (если есть), в блок в котором есть обращение к адресу, как используется адрес (для записи или для чтения) и зависящая от языка программирования информация. Для STL-программы столбец, зависящий от языка программирования, содержит информацию о сети, в которой адрес используется; для SCL-программы - содержит номер строки и столбца. Щелкните на заголовке столбца для сортировки таблицы по содержанию столбца.

Если адрес выбран, Вы с помощью выбора опций меню: *Edit -> Go To -> Line (Правка -> Перейти -> На строку)* можете запустить редактор программ, и при этом будет отображен фрагмент программы с выбранным адресом.

Список перекрестных ссылок можно просматривать с использованием фильтра (опции: *View -> Filter… (Вид -> Фильтр…)* для получения информации по интересующим адресам (например, проверить использование меркеров). Если дважды щелкнуть на адресе, будет в редакторе открыт блок на строке, в которой данный адрес используется. STEP 7 всякий раз при открытии списка перекрестных ссылок использует сохраненную (как "Standard") настройку фильтра.

Польза от применения: перекрестные ссылки показывают, были ли адреса использованы для считывания или сброса сигнала. Также они показывают, в каких блоках адреса использованы (возможно, больше одного раза).

Назначения (Assignments)

Список ссылок на I/Q/M показывает, какие биты в адресных областях I, Q и M имеют назначения в программе. В каждой строке показывается побитно один байт. Также указывается тип доступа (побайтный, пословный или двухсловный). Список ссылок на T/C показывает таймеры и счетчики, использованные в программе. В строке показываются по десять таймеров или счетчиков.

Польза от применения: список ссылок показывает, были ли определенные адресные области назначены (заняты) или какие адреса остаются все еще доступными.

Структура программы (Program structure)

Структура программы показывает иерархию вызовов блоков в пользовательской программе. Блоки вызова ("starting blocks") в иерархии вызовов выбираются с помощью настроек фильтров. Вы можете выбирать из двух различных видов:

"Древовидная" структура (tree structure) показывает все уровни вложения в системе вызовов блоков. Пользователь может изменять отображение уровней вложения, используя элементы управления в структуре - квадраты со значками "+" и "-". Требования, предъявляемые к временным локальным данным ("temporary local data"), показаны в целом для цепочки ("path") вызовов. Щелкните правой кнопкой мыши в поле разворачивающегося меню для открытия соответствующего блока, перейдите к месту вызова или к экрану с дополнительной информацией по блоку.

Структура "Родитель-потомок" (Parent-child structure) показывает 2 уровня с одним вызываемым блоком, а также информацию, зависящую от языка.

Польза от применения: структура программы показывает, какие блоки используются, все ли запрограммированные блоки вызываются, какие требования предъявляются к временным локальным данным, удовлетворяют ли локальные данные определенным требованиям для приоритетного класса (для организационного блока).

Неиспользуемые символы (Unused symbols)

Данный список показывает все адреса, имеющие назначения в таблице символов, но не используются в программе. Список показывает символ, адрес, тип данных и комментарий из таблицы символов.

Польза от применения: список позволяет определить, не были ли отдельные адреса по невнимательности забыты при записи программы, или, может быть, они избыточны и в действительности не нужны.

Адреса без символов (Addresses without symbols)

Список показывает все используемые в программе адреса, которые не имеют символов. Список показывает сами адреса, и как часто они используются.

Польза от применения: список позволяет определить, не используются ли отдельные адреса по невнимательности, (по случайности или по ошибке).

2.5.8 Многоязыковая поддержка комментариев и отображаемых текстов

Утилита SIMATIC Manager позволяет пользователю управлять несколькими версиями языка для комментариев и отображаемых текстов.

Выбор языка для комментариев и отображаемых текстов определяет язык текстов, вводимых пользователем. Язык сессии, заданный, например, для меню, сообщений об ошибках, устанавливается вместе с STEP 7 и выбирается с помощью утилиты SIMATIC Manager в опциях меню: *Options -> Customize (Опции -> Установки пользователя)* на вкладке "Language" ("Язык"). Здесь же Вы можете задать мнемоники, т.е. язык операторов и операндов, используемый в STEP 7. Все три установки не зависят друг от друга.

Общая процедура

Допустим, Вы ввели тексты на языке оригинала, (например, на английском языке), и Вам необходимо получить версию Вашей программы на немецком языке. Чтобы сделать это, экспортируйте требуемые тексты или типы текстов. Например, экспортируемый файл является *.csy файлом, который Вы можете редактировать с помощью Microsoft Excel. Вы можете ввести перевод (translation) для каждого текста и затем импортировать законченную таблицу трансляции (translation table) обратно в Ваш проект. Теперь Вы можете переключать языки текстов (сообщений) в проекте. У Вас есть возможность использовать для этого несколько языков.

Экспорт и импорт текстов (Exporting and importing texts)

С помощью утилиты SIMATIC Manager выберите объект, содержащий комментарии, которые необходимо перевести (транслировать), например, таблицу символов, каталог блоков (block container), несколько блоков или отдельный блок. Выберите опции меню: *Options -> Manage Multilingual Text -> Export (Опции -> Многоязыковая поддержка текста -> Экспорт).*

В появившемся окне диалога введите местоположение экспортируемого файла (storage location) и язык, на который требуется транслировать текст (target language). Выберите типы текстов (Text type), которые Вы хотите транслировать (Таблица 2.2).

Таблица 2.2

Типы транслируемых текстов (Text type) (извлечение)

Text type (Типы текста)	**Meaning (Значение)**
BlockTitle	Block title (Заголовок блока)
BlockComment	Block comment (Комментарий блока)
NetworkTitle	Network Title (Заголовок сегмента)
NetworkComment	Network Comment (Комментарий сегмента)
LineComment	Line Comment (Комментарий строки)
InterfaceComment	Comment in (Комментарий • declaration table of code blocks - таблицы объявлений блоков кода • data blocks - блоков данных • user data type UDT - пользовательских типов данных)
SymbolComment	Symbol Comment (Комментарий символа)

Для каждого типа текста генерируется отдельный файл, например, для комментариев из таблицы символов – файл SymbolComment.csv. Существующие экспортированные файлы могут быть расширены.

Откройте экспортированный(ые) файл(ы) в диалоговом окне Microsoft Excel с помощью опций меню: *File -> Open (Файл -> Открыть)* (Не открывать двойным щелчком на файле). Экспортированные тексты отображаются в первом столбце, и Вы можете ввести перевод данных текстов (транслировать) во втором столбце.

Вы можете вернуть переведенные тексты обратно в проект посредством опций меню: *Options -> Manage Multilingual Text -> Import (Опции -> Многоязыковая поддержка текста -> Импорт).* Файл протокола (log-файл) содержит информацию об импортированных текстах и о любых ошибках, которые могли произойти.

Примечание: Имя импортированного файла не может быть изменено, так как есть прямая связь между именем и типом текста (Text type), который содержится в файле.

Выбор и удаление языка

Вы можете переключать отображаемые тексты на любой инсталлированный в системе язык с помощью утилиты SIMATIC Manager в опциях меню: *Options -> Manage Multilingual Text -> Change Language (Опции -> Многоязыковая поддержка текста -> Сменить язык).* Процедура смены языка выполняется для объектов (блоков, таблицы символов), для которых соответствующие тексты были импортированы. Информация об этом содержится в файле протокола (в log-файле).

Вы можете также удалить любой инсталлированный в системе язык с помощью утилиты SIMATIC Manager в опциях меню: *Options -> Manage Multilingual Text -> Delete Language (Опции -> Многоязыковая поддержка текста -> Удалить язык).*

2.6 Интерактивный режим (Online Mode)

Пользователь создает конфигурацию оборудования и пользовательскую программу, в основном опираясь на технические средства “engineering system” (ES). S7-программа сохраняется автономно (offline) на жестком диске в текстовой форме, а также в скомпилированном виде.

Для переноса программы в CPU Вы должны соединить программатор PG с CPU, после чего - установить интерактивное ("online") соединение. Вы можете использовать это соединение для определения рабочего состояния CPU и назначенных (assigned) модулей, т.е. Вы можете реализовать функции диагностики.

2.6.1 Подключение к PLC (Connection a PLC)

Соединение MPI-интерфейса программатора PG и MPI-интерфейса CPU является аппаратным требованием для интерактивного (online) соединения. Такое соединение является единственным, если CPU является единственным подключенным программируемым модулем. Если существует несколько CPU в MPI-подсети, то каждому CPU должен быть назначен уникальный номер узла (MPI-адрес). Вы должны установить MPI-адрес при инициализации CPU. Перед связыванием всех CPU в единую подсеть подключите программатор только к одному CPU и перешлите туда объект *System Data (Системные данные)* из автономного каталога *Blocks (Блоки)* или непосредственно из утилиты конфигурирования Hardware Configuration с помощью опций меню: *PLC -> Download (PLC -> Загрузить).* При этом CPU получит свой собственный специальный MPI-адрес ("naming" - наименование) вместе с другими характеристиками.

MPI-адрес CPU в MPI-сети может быть изменен в любое время с помощью пересылки записи данных с новыми параметрами, содержащими новый MPI-адрес в CPU.

Будьте внимательны: изменение MPI-адреса имеет мгновенный эффект.

Программатор немедленно настраивается на новый адрес, поэтому необходимо настроить остальные параметры, такие, например, как коммуникации посредством глобальных данных на новый MPI-адрес.

MPI-параметры остаются в CPU даже после сброса памяти. Таким образом, CPU может быть адресован даже после сброса памяти.

Во всех случаях программатор может быть задействован в интерактивном режиме (online) с CPU, даже при использовании программы, независящей от модулей, и даже если в PG не установлено соответствующего проекта.

Если в PG не установлено соответствующего проекта, Вы должны установить соединение с CPU с помощью опций меню: *PLC -> Display Accessible Nodes (PLC -> Отобразить доступные узлы)*. При этом будет показано окно проекта (project) со структурой:

"*Accessible Nodes*" - "Module (MPI=n)" - "Online User Program (*Blocks*)" ["*Доступные узлы*" - "Модуль (MPI=n)" - "Интерактивная программа пользователя (*Блоки*)"].

При выборе объекта *Module* Вы можете использовать интерактивные (online) функции, такие как изменение рабочего состояния (operational status) и проверка состояния модулей (module status). Если выбрать объект *Blocks (Блоки),* то будут отображаться блоки, находящиеся в пользовательской памяти в CPU. При этом Вы можете редактировать (изменять, удалять, вставлять) отдельные блоки.

Вы можете считать системные данные из подключенного CPU с целью, скажем, последующей работы с существующей конфигурацией в отсутствие соответствующего проекта в системе управления данными PG (data management system).

Для этого создайте новый проект в SIMATIC Manager, выберите проект, затем - опции меню: *PLC -> Upload Station (PLC -> Считать конфигурацию станции).* После определения требуемого CPU в появившемся окне диалога интерактивные (online) системные данные записываются на жесткий диск.

Если программа не зависит от CPU, Вы должны включить окно проекта для интерактивного (online) режима. Если в MPI-подсети имеется несколько CPU и они доступны, то выбрав интерактивную (online) S7-программу и опции меню: *Edit -> Object Properties (Правка -> Свойства объекта),* задайте номера монтажных стоек и слотов установки CPU в таблице адресации модулей "Addresses Module".

Если Вы выбрали *S7-программу* в окне проекта для интерактивного (online) режима, то Вам будут доступны все интерактивные (online) функции для подключенного CPU. *Blocks (Блоки)* показывает блоки, размещенные в пользовательской памяти CPU. Если блоки в автономной (offline) программе соответствуют блокам в интерактивной (online) программе, то Вы можете редактировать блоки в пользовательской памяти, используя информацию системы управления данными программатора PG (символьные адреса, комментарии).

Если Вы переключаете **программу, предназначенную для CPU**, в интерактивный (online) режим, Вы можете делать изменения в программе только, если Вы открыли независимую от CPU программу. Вдобавок, теперь Вы можете конфигурировать SIMATIC-станцию, т.е. установить параметры CPU, а также адресовать и параметризировать модули.

2.6.2 Защита программы пользователя

Для соответствующим образом оснащенных CPU доступ к программе пользователя может быть защищен паролем. Каждый, кто знает пароль, имеет неограниченный доступ к пользовательской программе. Для тех, кто не знает пароля, Вы можете установить три уровня (степени) защиты. Установка уровней защиты производится на вкладке "Protection" ("Защита") в соответствующем окне утилиты для конфигурирования оборудования Hardware Configuration при параметризации CPU.

Уровень защиты 1: блокировка положения переключателя
(Protection level 1: keylock switch position)

Данный уровень защиты устанавливается по умолчанию (если не введен пароль). В этом случае программа пользователя защищена блокировкой функций переключателя режимов на передней панели CPU. В положениях RUN-P и STOP Вы имеете неограниченный доступ к программе пользователя; в позиции RUN возможен доступ в режиме "только чтение" с использованием программатора PG. Для этой позиции Вы можете также удалять блокировку, так чтобы режим не мог быть в дальнейшем изменен с помощью переключателя.

Вы можете обходить защиту посредством блокировки положения RUN, выбрав опцию, "Can be revoked with password" ("Может быть отменено при вводе пароля"), например, если CPU и его заблокированный переключатель режима находятся далеко или в трудно доступном месте.

Уровень защиты 2: защита от записи
(Protection level 2: write protection)

Данный уровень защиты обеспечивает доступ к программе пользователя только в режиме чтения вне зависимости от положения переключателя режимов.

Уровень защиты 3: защита в режиме чтения и записи
(Protection level 3: read/write protection)

Данный уровень защиты полностью блокирует доступ к программе пользователя вне зависимости от положения переключателя режимов.

Защита паролем (Password protection)

Если Вы выбираете уровни защиты 2 или 3 или уровень защиты 1 с опцией "Can be revoked with password" ("Может быть отменено при вводе пароля"), то Вам будет предложено определить пароль. Пароль может иметь размер до 8 символов.

При попытке получить доступ к программе пользователя, защищенной паролем, пользователю будет предложено ввести пароль. Если необходимо получить доступ к защищенному паролем CPU, пользователь также может ввести пароль посредством опций меню: *PLC -> Access Rights (PLC -> Права доступа)*. Необходимо сначала выбрать CPU или S7-программу.

В диалоговом окне "Enter Password" ("Введите пароль") Вы можете выбрать опцию "Use password for other protected modules" ("Использовать пароль для других защищенных модулей") для предоставления доступа ко всем защищенным модулям с одинаковым паролем.

При этом доступ с паролем авторизации ("Password access autorization") остается в силе до тех пор, пока последнее приложение системы S7 не будет деинсталлировано.

Каждый, кто знает пароль, имеет неограниченный доступ к пользовательской программе в CPU независимо от установленного уровня защиты и от блокировки переключателя режимов.

2.6.3 Информация о CPU (CPU Information)

При интерактивном (online) режиме Вам доступна представленная ниже информация, касающаяся CPU. Команды меню выводятся на экран, если Вы выбираете какой-либо модуль (в интерактивном [online] режиме и без проекта) или S7-программу (в интерактивном [online] окне проекта).

- *PLC -> Diagnose Hardware (PLC -> Диагностика оборудования)*

(см. раздел 2.7.1 "Диагностика оборудования")

- *PLC -> Module Information (PLC -> Информация о модуле)*

Общая информация (такая, например, как версия), диагностический буфер, память (текущее распределение [map] рабочей памяти [work memory] и загрузочной памяти [load memory], сжатие [compression]), время цикла [cycle time] (длина последнего цикла, длина самого длинного и длина самого короткого цикла программы), система времени (внутренние часы CPU, синхронизация времени, счетчик рабочего времени), рабочие характеристики (конфигурация памяти, размеры адресных областей, число доступных блоков, SFC и SFB), коммуникации (скорость передачи данных и коммуникационные соединения), стеки в режиме STOP (B-стек, I-стек и L-стек) в таблице адресации модулей "Addresses Module".

- *PLC -> Operating Mode (PLC -> Рабочий режим)*

Отображение рабочего режима (например, RUN или STOP), изменение рабочего режима.

- *PLC -> Clear/Reset (PLC -> Очистка/Сброс)*

Сброс CPU в STOP-режиме.

- *PLC -> Set Date and Time (PLC -> Установить дату и время)*

Установка внутренних часов CPU.

- *PLC -> CPU Messages (PLC -> Сообщения CPU)*

Сообщения об асинхронных системных ошибках и сообщения, определяемые пользователем, генерируются в программе с помощью SFC 52 WR_USMSG, SFC 18 ALARM_S и SFC 17 ALARM_SQ.

- *PLC -> Display Force Values, (PLC -> Отобразить форсированные значения),*

  PLC -> Monitor/Modify Variables, (PLC -> Мониторинг/модификация переменных)

(см. раздел 2.7.3 "Мониторинг/модификация переменных" и раздел 2.7.4 "Форсирование переменных").

2.6.4 Загрузка пользовательской программы в CPU

При пересылке в CPU Вашей пользовательской программы (а именно, скомпилированных блоков и данных конфигурации) она загружается в загрузочную (load) память CPU. Физически загрузочная (load) память может быть построена на элементах RAM или флэш EPROM, она может быть встроенной в CPU или быть в виде отдельного модуля.

Если модуль памяти типа флэш EPROM, то он может быть использован как перемещаемый носитель информации: данные на него могут быть записаны с помощью программатора PG, после чего этот модуль может быть вставлен в CPU, находящийся в выключенном состоянии. После включения питания и последующего сброса памяти соответствующие данные из модуля памяти пересылаются в рабочую (work) память CPU. Некоторые типы CPU позволяют перезаписывать вставленные в них модули флэш EPROM-памяти, но при этом перезаписывается только программа в целом.

Если загрузочная (load) память имеет тип RAM, то перенос целиком программы пользователя в CPU осуществляется следующим образом: CPU переводится в состояние STOP, затем выполняется сброс памяти и пересылается программа пользователя. Пересылаются также конфигурационные данные. Программа в RAM-памяти стирается при сбросе памяти или если отключается резервная батарея питания (backup battery).

Если Вы хотите изменить только данные конфигурации (свойства CPU, сконфигурированные соединения, GD-коммуникации, параметры модулей и т.п.), то Вам нужно загрузить в CPU только объект *System data (Системные данные)*. Для этого выберите объект и перешлите его с помощью опций меню: *PLC -> Download (PLC -> Загрузить)*. Параметры CPU вступают в силу немедленно; параметры для остальных модулей пересылаются в эти модули во время запуска (startup).

Нужно иметь в виду, что в PLC полная конфигурация загружается с объектом *System data (Системные данные)*. Если Вы используете опции меню: *PLC -> Download (PLC -> Загрузить)* в приложении, например, в GD-коммуникациях, то только отредактированные в приложении данные будут пересланы в PLC.

Примечание: для загрузки сжатого архивного файла используйте опции меню: *PLC -> Save Project on Memory Card (PLC -> Сохранить проект в модуле памяти)* (см. раздел 2.2.2 "Управление, перекомпоновка и архивация"). Проект, находящийся в заархивированном состоянии, не может быть отредактирован непосредственно ни с помощью программатора PG, ни в CPU.

2.6.5 Работа с блоками (Block Handling)

Перенос блоков

Если загрузочная (load) память имеет тип RAM, то Вы можете не только переносить программу целиком в интерактивном режиме (online), но также изменять, удалять или перезагружать отдельные блоки.

Вы можете пересылать отдельные блоки в CPU, выбрав их в "автономном" (offline) окне с последующим использованием опций меню: *PLC -> Download (PLC -> Загрузить).* Открыв одновременно "интерактивное" (online) и "автономное" (offline) окна, Вы можете с помощью манипулятора "мышь" перетаскивать блоки из одного окна в другое (метод "drag-n-drop").

Особое внимание необходимо при пересылке отдельных блоков во время рабочего режима. Если блоки, которые не доступны в памяти CPU, вызываются внутри блока, Вы должны сначала загрузить блоки "нижнего уровня". Это также касается блоков данных, адреса которых используются в загружаемом блоке. Вы должны загрузить блок "высшего уровня" последним. Затем, будучи вызванным, он будет незамедлительно выполнен в следующем программном цикле.

Утилита SIMATIC Manager позволяет Вам также пересылать отдельные блоки или программу в целом из "автономного" (offline) каталога "Blocks" ("Блоки") в CPU на языке SCL. Обратный перенос из CPU на жесткий диск скомпилированных блоков не имеет никакого смысла, так как они не могут быть напрямую отредактированы в редакторе SCL-программ. Вы можете редактировать только исходную SCL-программу и после этого вновь из нее создавать скомпилированные блоки.

Изменение блоков в интерактивном (online) режиме

Вы можете инкрементно редактировать на языке STL блоки программы пользователя в интерактивном (online) режиме (то есть в CPU), точно таким же образом, как и программу в автономном (offline) режиме. Однако, если "интерактивная" и "автономная" программы пользователя отличаются друг от друга, то это может привести к тому, что редактор не сможет отобразить дополнительную информацию из "автономной" базы данных; при этом эти данные (символьные идентификаторы, метки перехода, комментарии, пользовательские типы данных) могут быть потеряны.

Блоки, которые необходимо изменять в интерактивном режиме, лучше хранить автономно (offline) на жестком диске для того, чтобы избежать потери консистентности данных (например, конфликт "несовпадение временных меток" ["time stamp conflict"], когда интерфейс вызываемого блока отстает от программы в вызывающем блоке).

Удаление блоков

Если загрузочная (load) память построена исключительно на RAM - элементах, то блоки можно изменять или удалять.

Если пользовательская программа расположена в модуле памяти типа флэш EPROM, то блоки могут также быть изменены или удалены при условии, что доступный объем дополнительной RAM-памяти достаточен. Блоки в модуле памяти типа флэш EPROM выделяются как "invalid" ("неправильные"). Тем не менее, после сброса памяти или после включения питания без резервной батареи блоки вновь пересылаются из загрузочной (load) памяти на модуле типа флэш EPROM в рабочую (work) память.

Вы можете удалять модуль памяти типа флэш EPROM только в программаторе PG.

Сжатие (compressing)

Если Вы загружаете новый или измененный блок в CPU, CPU помещает блок в загрузочную (load) память и передает релевантные данные в рабочую (work) память. Если в рабочей (work) памяти уже находится блок с таким же номером, то этот "старый блок" объявляется неправильным (invalid) (пользователю предлагается это подтвердить), и после этого новый блок добавляется в "конец" занятой области памяти (после последней записи). Если даже блок удален, он помечается как неправильный (invalid), но в действительности из памяти не удаляется.

Это приводит к разрывам (gap) в непрерывном занятом пространстве памяти и уменьшает общий объем доступной памяти. Эти разрывы могут быть заполнены только при использовании функции сжатия *Compress*. При использовании этой функции в режиме RUN блоки, выполняющиеся в текущий момент, не могут быть перемещены; только в режиме STOP Вы можете эффективно ликвидировать разрывы в непрерывном занятом пространстве памяти.

Текущее состояние загрузки памяти может быть отображено в процентах с помощью опций меню: *PLC -> Module Information (PLC -> Информация о модуле)* на вкладке "Memory" ("Память"). В появившемся диалоговом окне Вы можете включить функцию сжатия.

С помощью SFC 25 COMPRESS Вы можете запрограммировать запуск функции сжатия, управляемый событием.

Блоки данных в интерактивном (online) и автономном (offline) режимах

Адресам данных в блоке данных могут быть назначены начальные значения (*initial value*) и фактические значения (*actual value*) (см. раздел 3.6 "Программирование блоков данных"). Если блок данных загружен в CPU, то начальные (initial) значения пересылаются в загрузочную (load) память, а фактические значения (actual) пересылаются в рабочую (work) память. Каждое изменение значения в соответствии с адресацией данных в программе предопределяет изменение соответствующего фактического значения.

Если Вы выгружаете блок данных из CPU, значения данных блока берутся из рабочей (work) памяти, в которой хранятся все фактические (actual) значения данных. Вы можете наблюдать фактические (actual) значения данных во время их считывания с помощью опций меню: *View -> Data View (Вид -> Вид данных).* Если Вы модифицируете фактические (actual) значения данных в блоке данных, а затем вновь записываете блок в CPU, то модифицированные значения поступают в рабочую (work) память.

Если в качестве загрузочной (load) используется модуль памяти типа флэш EPROM, то блоки из модуля памяти переносятся в рабочую (work) память после сброса памяти CPU. При этом блок памяти сохраняет первоначально запрограммированные в нем значения данных. То же самое происходит при включении питания при отключенной резервной батарее. Для S7-300 Вы можете предотвратить загрузку первоначальных значений данных, если Вы объявите эти области данных реманентными (retentive)

Блок данных, созданный с активированным свойством "UNLINKED" ("несвязанный"), не переносится в рабочую (work) память; он остается в загрузочной (load) памяти. Блок данных с активированным свойством "UNLINKED" ("несвязанный") может быть считан только с помощью функции SFC 20 BLKMOV.

2.7 Тестирование программы

После выполнения соединения с CPU и загрузки пользовательской программы Вы можете тестировать (отлаживать) программу в целом или по частям, отдельными блоками. Необходимо инициализировать переменные значениями, определенными, например, с помощью модулей симулятора, и оценить информацию отклика, полученного программой в виде значений данных. Если в результате ошибки CPU переходит в состояние STOP, Вы можете использовать, в частности, информацию о CPU.

Большие программы обычно отлаживаются по частям. Если Вам, например, необходимо отлаживать один блок, то загрузите этот блок в CPU и вызовите его в организационном блоке OB1. Если блок OB1 построен таким образом, что программа может быть отлажена фрагмент за фрагментом от начала до конца, то Вы можете выбирать для отладки отдельные блоки или фрагменты программы, используя функции перехода, чтобы миновать разделы,

не нуждающиеся в отладке.

С помощью опционного (поставляемого по отдельному заказу) программного обеспечения PLCSIM, Вы можете моделировать CPU в программаторе PG и таким образом отлаживать Вашу программу без дополнительного оборудования.

2.7.1 Диагностика оборудования

В случае отказа Вы можете считать диагностическую информацию из отказавших модулей с помощью функции диагностики оборудования "Diagnose Hardware". Для этого Вам сначала необходимо подключить программатор PG к MPI-шине и запустить утилиту SIMATIC Manager.

Если проект, связанный с конфигурацией установки, доступен в базе данных программатора PG, то откройте интерактивное (online) окно проекта с помощью опций меню: *View -> Online (Вид - Интерактивный режим)*. В противном случае выберите опции: *PLC -> Display Accessible Nodes (PLC - Отобразить доступные узлы)* и затем выберите CPU.

Теперь Вы можете получить краткий обзор сбойных модулей с помощью опций: *PLC -> Diagnose Hardware (PLC - Диагностика оборудования)* (по умолчанию). Утилита конфигурирования оборудования Hardware Configuration поддерживает функцию детальной диагностической информации о модулях при интерактивном режиме; этот режим устанавливается в утилите SIMATIC Manager на вкладке "View" ("Вид") при выборе опций меню: *Options -> Customize (Опции -> Установки пользователя)*.

Вы можете получить информацию о состоянии (status) и рабочем состоянии (operating state) модулей, доступных в интерактивном режиме, в форме отображения проекта (project view - отображение станций проекта, сообщающих об ошибках), в форме отображения станции (station view - отображение модулей станции, сообщающих об ошибках) и в форме отображения модуля (module view - отображение соответствующей диагностической информации).

2.7.2 Определение причины перехода в состояние STOP

Если CPU переходит в состояние STOP из-за ошибки, первое, что нужно сделать для определения причины перехода в это состояние, - это вывести для чтения содержимое диагностического буфера. CPU вводит в диагностический буфер все сообщения, в том числе, сообщение о причине перехода в состояние STOP и сообщения об ошибках, которые привели к этому.

Для вывода содержимое диагностического буфера переключите программатор PG в интерактивный (online) режим, выберите S7-программу и активируйте вкладку *Diagnostics Buffer* (*Диагностический буфер*) с помощью опций меню: *PLC -> Module Information (PLC -> Информация о модуле)*. Последнее событие из буфера (первое событие имеет номер 1) и есть причина перехода CPU в состояние STOP, например, "STOP because programming error OB not loaded" ("Состояние STOP из-за ошибки программы - блок OB не загружен").

Ошибка, которая привела к переходу CPU в состояние STOP, описана в предыдущем сообщении, например: "FC not loaded" ("FC не загружен"). Щелчком на номере сообщения Вы можете вывести на экран дополнительный комментарий в следующем нижнем поле экрана. Если сообщение касается ошибок программирования в блоке, Вы сможете открыть и отредактировать тот блок, нажав кнопку "Open Block" ("Открыть блок").

Если, например, причиной перехода CPU в состояние STOP является ошибка программирования, Вы можете установить "обстоятельства окружения" фрагмента программы, содержащего ошибку, с помощью вкладки "Stacks" ("Стеки"). Когда Вы откроете вкладку "Stacks" ("Стеки"), Вы увидите B-стек (block stack - стек блоков), который показывает расположение вызова всех незавершенных блоков вплоть до блока, в котором находится точка прерывания. Используя кнопку "I stack", Вы получите данные стека прерываний (interrupt stack), показывающего содержание регистров CPU (аккумуляторов, адресного регистра, регистра блока данных, слово состояния) в точке прерывания в тот момент, когда произошла ошибка. Используя кнопку "L stack" (local data stack - стек локальных данных), Вы получите доступ к локальным данным блока, который можно выбрать в окне B-стека. Перейти к окну B-стека можно с помощью щелчка манипулятора "мышь" на соответствующей кнопке.

2.7.3 Мониторинг и модификация переменных (Monitoring and Modifying Variables)

Есть замечательное средство для отладки пользовательской программы - функция для мониторинга и модификации переменных (Monitoring and Modifying of Variables), использующая VAT-таблицу (таблицу размещения переменных). Состояния сигналов или значения переменных простых типов данных могут быть отображены с помощью этого средства. При наличии доступа к пользовательской программе Вы можете также модифицировать переменные, т.е. изменять состояния сигналов или назначать новые значения.

Предупреждение: Вы должны избегать опасных состояний в Вашей установке, могущих возникать при изменении значений переменных!

Создание таблицы переменных

Для того, чтобы использовать функцию для мониторинга и модификации переменных (Monitoring and Modifying of Variables), Вы должны создать VAT-таблицу (таблицу размещения переменных), содержащую переменные и форматы соответствующих данных. Вы можете генерировать до 255 таблиц переменных (VAT1 … VAT255) и назначить им имена в таблице символов (Symbol Table). Максимальный размер VAT-таблицы составляет 1024 строки с содержанием до 255 символов (см. рис. 2.11).

Вы можете создать VAT-таблицу автономно (offline), выбрав пользовательскую программу *Blocks (Блоки)*, а затем опции меню: *PLC -> Monitor/Modify Variables (PLC -> Мониторинг/модификация переменных)*.

Вы можете определять переменные с помощью абсолютной или символьной адресации и выбрать для них тип данных (формат отображения переменной).

Для изменения выберите строки, затем: *View -> Display Format (Вид -> Отобразить формат)*, или просто щелкните правой кнопкой мыши на заголовке столбца "Display Format" ("Отобразить формат").

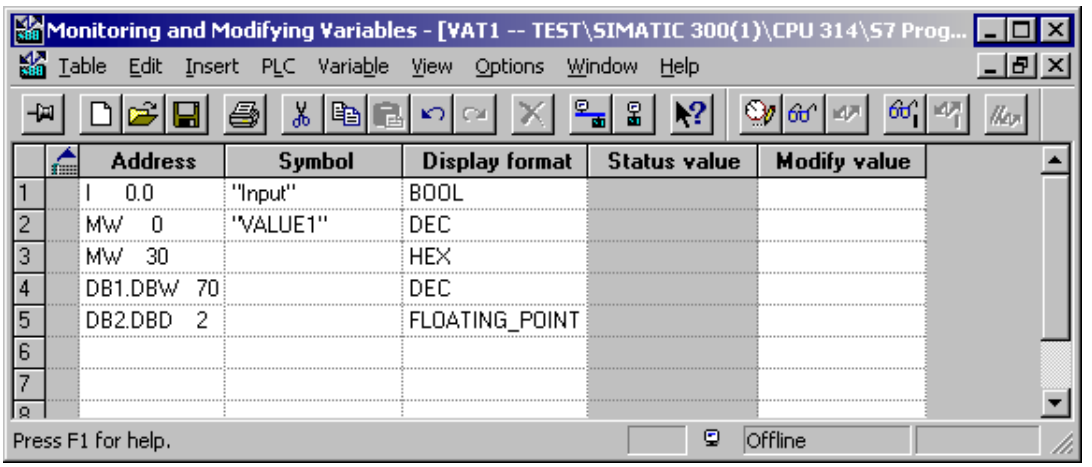


Рис. 2.11 Пример таблицы переменных (Variable Table)

Используйте строки комментариев для разделения таблицы на отдельные секции и придания отдельным частям таблицы заголовков. Вы можете также определять вид таблицы, а именно, какие столбцы должны быть отображены. В любое время Вы можете изменить переменные или формат их отображения, добавить или удалить строки таблицы. Таблица переменных должна быть сохранена в каталоге объекта *Blocks (Блоки)* с помощью опций: *Table -> Save (Таблица -> Сохранить)*.

Установление интерактивного (online) соединения

Для работы с VAT-таблицей, которая была создана автономно (offline), переключите ее в интерактивный режим с помощью опций меню: *PLC -> Connect To … (PLC -> Подключить к …)*. Вы должны переключать в интерактивный режим каждую таблицу отдельно, а после работы с таблицей - отключать это соединение с помощью опций: *PLC -> Disconnect (PLC -> Разъединить)* .

Условия запуска (Trigger conditions)

В таблице переменных выберите опции меню: *Variable -> Trigger (Переменная -> Запуск)* для установки точки запуска (trigger point) и условий запуска (trigger conditions) отдельно для функций мониторинга и модификации. Точка запуска (trigger point) - это такая точка, в которой CPU считывает значения из системной памяти или записывает значения в системную память.

Вы должны определить, будет ли считывание или запись происходить один раз или будет периодическим.

Если для функции мониторинга и модификации имеются одинаковые условия запуска, то функция мониторинга будет выполняться до функции модификации. Если Вы выбрали для функции модификации точку запуска "Start of cycle" ("Начало цикла"), то переменные будут модифицированы после обновления отображения входов процесса и перед вызовом блока OB 1. Если Вы выбрали для функции мониторинга точку запуска "End of cycle" ("Конец цикла"), то состояния переменных будут выведены после завершения OB1 и перед установкой выходов в соответствии с отображением выходов процесса.

Мониторинг переменных (Monitoring of Variables)

Для выбора функции мониторинга используйте опции меню: *Variable -> Monitor (Переменная -> Мониторинг)*. Переменные из VAT-таблицы обновляются в соответствии с определенными условиями запуска. Постоянный мониторинг позволит Вам следить за изменениями значений переменных на экране. Значения отображаются в формате данных, который Вы задаете в столбце Display Format (Формат отображения). Закончить непрерывный процесс мониторинга позволяет кнопка Esc.

Использование опций: *Variable -> Update Monitor Values (Переменная -> Обновить отслеживаемые значения)* позволяет однократно обновить наблюдаемые переменные, причем это обновление происходит мгновенно и не зависит от заданных условий запуска.

Модификации переменных (Modifying of Variables)

Для выбора функции модификации переменных (для пересылки в CPU измененных значений переменных) в соответствии с заданными условиями запуска используйте опции меню: *Variable -> Modify (Переменная -> Модификация)*. Вводите модифицированные значения только в те строки VAT-таблицы, в которых содержатся переменные, которые нужно изменить. Вы можете распространить комментарий на значение ("закомментировать" значение переменной) с помощью символов "//" или с помощью опций: *Variable -> Modify Value Valid (Переменная -> Модификация значения разрешена)*; такие значения не берутся в расчет при модификации переменных. Значения отображаются в формате данных, который Вы задаете в столбце Display Format (Формат отображения). Закончить непрерывный процесс модификации можно с помощью кнопки Esc.

Использование опций: *Variable -> Activate Modify Values (Переменная -> Активировать модификацию значений)* позволяет однократно модифицировать переменные, причем это происходит мгновенно и не зависит от заданных условий запуска.

2.7.4 Форсирование переменных (Forcing Variables)

Отдельные типы CPU позволяют использовать особую функцию - форсирование переменных (Forcing Variables), заключающуюся в том, что Вы с ее помощью можете задавать фиксированные значения некоторым переменным.

При этом пользовательская программа не сможет изменить эти значения. Форсирование разрешено для любого режима CPU и выполняется немедленно после запуска функции.

Предупреждение: Вы должны избегать опасных состояний в Вашей установке, могущих возникнуть при форсировании значений переменных!

Отправной точкой для форсирования переменных является VAT-таблица. Вы должны создать VAT-таблицу, после этого - задать адреса, для которых требуется форсирование значений. Затем необходимо установить соединение с CPU. Вы можете открыть окно, содержащее форсируемые значения, выбрав опции меню: *Variable -> Display Force Values (Переменная -> Отобразить форсированные значения).*

Если форсированные значения уже активны в CPU, это отображается в окне функции форсирования (force window) с помощью выделенного шрифта. Вы можете теперь перенести некоторые или все адреса из таблицы переменных в окно функции форсирования или внести в этом окне новые адреса. После определения переменных для форсирования значений Вы должны сохранить содержание окна функции форсирования с помощью опций меню: *Table -> Save As (Таблица -> Сохранить как).*

Функция форсирования значений может быть использована для следующих адресных областей:

- Входы I (отображение процесса)

  [S7-300 и S7-400]
- Выходы Q (отображение процесса)

  [S7-300 и S7-400]
- Периферийные входы PI

  [только S7-400]
- Периферийные выходы PQ

  [S7-300 и S7-400]
- Меркеры M

  [только S7-400]

Вы можете запустить функцию форсирования с помощью опций меню: *Variable -> Force (Переменная -> Активировать форсирование значений).* CPU использует форсированные значения для заданных переменных и не разрешает в дальнейшем изменять значения этих переменных.

Пока активна функция форсирования:

- Все попытки чтения по адресу форсированной переменной из пользовательской программы (например, load [загрузить]) и из системной программы (например, обновление образа процесса) всегда оканчиваются с одним результатом: величина переменной соответствует форсированному значению.
- В S7-400 все попытки записи по адресу форсированной переменной из пользовательской программы (например, transfer [переслать]) и из системной программы (например, посредством SFC) всегда оканчиваются без результата: изменения переменной запрещены. В S7-300 из пользовательской программы можно изменить ранее форсированное значение переменной.

Функция форсирования переменных в S7-300 соответствует функции модификации в циклическом режиме: после обновления отображения входов процесса CPU перезаписывает входы форсированными значениями; перед установкой выходов процесса в соответствии с отображением выходов процесса CPU перезаписывает последние форсированными значениями.

Примечание: функция форсирования не завершается с закрытием окна функции форсирования, таблицы переменных или при разрыве связи с CPU!

Остановить работу функции форсирования переменных можно, если только Вы используете опции меню: Variable -> Delete Force (Переменные -> Отменить функцию форсирования).

Функция форсирования также может быть остановлена, если выполнить сброс памяти или выключить (перевключить) питание, при условии, что CPU не имеет резервной батареи питания.

Если функция форсирования остановлена, соответствующие адреса продолжают содержать форсированные значения до тех пор, пока они не будут изменены или из пользовательской, или из системной программы.

Функция форсирования имеет стабильный эффект только для изменения I/O в CPU. Если после перезапуска форсированные PI и PQ больше не назначаются (например, в результате новой параметризации), то эти PI и PQ не поддерживают форсированные значения.

Обработка ошибок

Если при считывании оказывается, что "ширина доступа" (access width) больше, чем размер форсируемых данных (например, форсируется байт [byte] в слове [word]), то не форсируемая часть значения адреса считывается как обычно. Если при этом происходит ошибка синхронизации (ошибка доступа или ошибка длины данных [access or area length error]), то программой пользователя или CPU фиксируется "ошибка вставки значения" ["error substitute value"] или же CPU переходит в состояние STOP.

Если при записи оказывается, что "ширина доступа" (access width) больше, чем размер форсируемых данных (например, форсируется байт [byte] в слове [word]), то не форсируемая часть значения адреса записывается как обычно. При подобной ошибке доступа при записи форсированный компонент адреса остается неизменным, то есть защита от записи (write protection) не отменяется ошибкой синхронизации (synchronization error).

Считывание (loading) форсированных периферийных выходов дает в результате форсированные значения. Если "ширина доступа" (access width) соответствует размеру форсируемых данных, входные модули, которые вставляются в стойку взамен отказавших или для расширения, могут получить форсированные значения.

Вход I в образе процесса, связанный с форсированным периферийным входом PI, не форсируется; заранее он не определен и может быть переопределен. При обновлении образа процесса данный вход получает форсированное значение периферийного входа.

При форсировании периферийных выходов PQ связанный выход Q в образе процесса не обновляется и не форсируется (форсирование действует только "внешне" ["externally"] на выходы модуля). Значения выходов сохраняются и могут быть перезаписаны; считывание с выходов показывает записанные значения (не форсированные значения). Если выходной модуль форсирован,

и если потом этот модуль отказал или удален, то он будет вновь принимать форсированные значения, когда он будет вновь включен в стойку в работоспособном состоянии.

Выходные модули выводят состояние сигнала "0" или предустановленное значение (substitute value) по OD сигналу (блокировка выходных модулей в режимах STOP [стоп], HOLD [пауза] и RESTART [перезапуск]) - даже если периферийные выходы форсированы (исключение составляют аналоговые модули без распознавания сигнала OD, которые продолжают выдавать на выход форсированное значение сигнала). Если сигнал OD выключен, функция форсирования вновь продолжает действовать.

Если в режиме STOP активирована функция *Enable PQ (Разблокировать PQ)*, то форсированные значения также имеют эффект в режиме STOP (благодаря деактивации OD-сигнала). Кода действие функции *Enable PQ (Разблокировать PQ)* прекращается, модули вновь переходят в безопасное ("safe") состояние (состояние сигнала "0" или предустановленное значение [substitute value]); при этом форсированное значение выхода вновь становится действительным при переходе в режим RUN.

2.7.5 Разблокировка периферийных выходов (функция Enable peripheral outputs)

В режиме STOP выходные модули обычно заблокированы OD-сигналом. С помощью функции "Enable peripheral outputs" ("Разблокировка периферийных выходов") Вы можете деактивировать OD-сигнал, таким образом, Вы сможете модифицировать сигналы выходных модулей даже в случае, если CPU находится в режиме STOP. Модификация сигналов производится с помощью таблицы переменных. Модифицируются только назначенные CPU периферийные выходы. Возможное применение для этого: тестирование монтажа выходов в STOP-режиме без пользовательской программы.

Предупреждение: Вы должны предотвратить возможность возникновения опасных ситуаций при разблокировке периферийных выходов.

Создайте таблицу переменных и введите в нее периферийные выходы (PQ), теперь модифицируйте значения переменных. Переключите таблицу переменных в интерактивный (online) режим с помощью опций меню: *PLC -> Connect To (…,* и остановите CPU, если необходимо, например, с помощью опций меню: *PLC -> Operating Mode*, затем выберите "STOP".

Вы можете деактивировать OD-сигнал с помощью опций меню: *Variable -> Enable Peripheral Outputs (Переменные -> Разблокировка периферийных выходов);* выходы модуля имеют состояние сигнала "0", или предустановленное значение (substitute value), или форсированное значение (force value). Вы можете модифицировать периферийные выходы с помощью опций меню: *Variable -> Activate Modify Values (Переменные -> Активировать изменение выходов).* Вы можете изменить значения для модификации значений переменных и вновь после этого активизировать функцию модификации.

Вы можете деактивировать функцию модификации периферийных выходов с помощью опций меню: *Variable -> Enable Peripheral Outputs (Переменные -> Разблокировка периферийных выходов)* или с помощью кнопки ESC.

При этом вновь активируется OD-сигнал, и выходы модуля принимают состояние сигнала "0", или предустановленное значение (substitute value), или форсированное значение (force value) сбрасывается.

Если режим STOP деактивируется в то время, пока активна функция "Enable peripheral outputs" ("Разблокировка периферийных выходов"), все периферийные входы сбрасываются, OD-сигнал активируется при переходе к режиму перезапуска (RESTART), а затем вновь активируется при переходе к режиму RUN.

2.7.6 Функция "Program Status" ("Состояние программы") для STL

С помощью функции "Program Status" ("Состояние программы") программный редактор обеспечивает дополнительные возможности для тестирования пользовательской программы. С помощью этой функции редактор отображает последовательно для каждой строки программы состояние выбранного Вами регистра. Настройка отображаемых данных выполняется на вкладке "STL" после выбора опций меню: *Options -> Customize (Опции -> Установки пользователя)* ("Standard" ["Стандарт"] предполагает отображение аккумулятора 1 или значений таймера или счетчика).

Для отладки блок программы помещается в пользовательскую память (user memory) CPU, вызывается и обрабатывается в редакторе. Откройте этот блок, например, с помощью двойного щелчка кнопки манипулятора "мышь" на блоке в интерактивном (online) окне SIMATIC Manager. Редактор запускается и отображает программу, содержащуюся в блоке.

Выберите подсеть, которую необходимо отладить. С помощью опций меню: *Debug -> Monitor (Отладка -> Мониторинг)* активируйте функцию Program Status (Состояние программы). Теперь Вы можете наблюдать состояния адресов, результат логической операции и назначения регистров. С помощью опций меню: *Debug -> Monitor (Отладка -> Мониторинг)* Вы можете также деактивировать функцию Program Status (Состояние программы).

С помощью опций меню: *Debug -> Call Environment (Отладка -> "Обстоятельства вызова")* Вы можете задать условия запуска функции. Такие установки Вам потребуются, если блок, который Вы отлаживаете, вызывается более чем один раз в Вашей программе. Вы можете инициировать запись состояний (status recording), или определив порядок вызовов, или сделав его зависимым от открытого блока данных. Если блок вызывается только один раз, то выберите "No Condition" ("Нет условий").

Вы можете модифицировать переменные, используя функцию Program Status (Состояние программы). Выберите адрес данных, которые должны быть модифицированы, затем выберите опции: *Debug -> Modify Address (Отладка -> Модифицировать адрес)*.

Запись информации функции Program Status (Состояние программы) требует дополнительного времени в цикле выполнения программы. Поэтому Вы можете выбирать один из двух рабочих режимов для отладки программы: "debug mode" (режим отладки) и "process mode" (режим обработки процесса). Первый из указанных режимов (режим "debug mode" [режим отладки]) позволяет использовать все функции отладки без ограничения. Этот режим выбирается, например, для отладки блоков без подключения к системе, так как отдельные функции отладки сильно увеличивают время выполнения цикла сканирования программы.

В режиме "process mode" (режим обработки процесса) необходимо стремиться минимизировать удлинение цикла сканирования из-за функций отладки, поэтому здесь накладываются определенные ограничения, например, на программные циклы (не все проходы циклов программы отображаются). Отдельные CPU позволяют установку рабочего режима на вкладке "Protection" ("Защитный режим") при параметризации CPU. Если при параметризации CPU был установлен "debug mode" (режим отладки), то Вы сможете изменить режим работы только во время повторной процедуры параметризации. Другими словами, режим может быть изменен в диалоговом окне. Установка рабочего режима отображается с помощью опций: *Debug -> Operation (Отладка -> Работа).*

Функции отладки Breakpoints (точки прерывания), Single-step Mode (пошаговый режим)

Для блоков, написанных на языке STL, некоторые CPU позволяют выполнять отладку программы оператор за оператором в пошаговом режиме "Single-step mode". CPU находится в режиме HOLD; в целях безопасности периферийные выходы заблокированы. Используя точки прерывания (breakpoints), Вы можете остановить программу в любой момент и отлаживать в пошаговом режиме (step-by-step).

Должен быть установлен режим "debug mode" (режим отладки). Если при параметризации CPU был установлен "debug mode" (режим отладки), то Вы можете изменить режим работы только во время повторной процедуры параметризации. Другими словами, режим может быть изменен в диалоговом окне.

Для установки точки прерывания (breakpoint) установите курсор в строке с нужным оператором и выберите опции: *Debug -> Set Breakpoint (Отладка -> Установка точки прерывания)*. Для отладки выберите опции меню: *Debug -> Breakpoints Active (Отладка -> Активировать точки прерывания)*; эта команда вызовет перенос точек прерывания в CPU и активирует их. CPU необходимо перезапустить, в процессе обработки программы CPU при достижении точки прерывания переходит в состояние паузы HOLD. При этом в специальном окне будет отображено текущее содержание регистра.

Теперь Вы можете запустить программу для выполнения в построчном (в пошаговом) режиме, выбрав опции меню: *Debug -> Execute Next Statement (Отладка -> Выполнить следующий оператор)*. При этом выполнение программы будет прерываться на каждом операторе, и при этом будет отображаться текущее содержание регистра. Если в текущем операторе производится вызов блока, Вы можете продолжить пошаговую обработку программы в этом блоке, если выберете опции: *Debug -> Execute Call (Отладка -> Выполнять вызов)*.

С помощью опций меню: *Debug -> Resume (Отладка -> Продолжить)* программа выполняется с нормальной скоростью, пока не встретит следующую точку прерывания.

Блоки, содержащие точки прерывания, не могут быть изменены или перезагружены в интерактивном режиме (online). Все точки прерывания сначала должны быть удалены. Также все точки прерывания должны быть удалены при выходе из режима отладки с точками прерывания. С помощью опций меню: *Debug -> Resume (Отладка -> Продолжить)* CPU снова переключается в режим RUN.

2.7.7 Отладка SCL-программ

Если необходимо отладить SCL-программу, Вы должны скомпилировать ее с опцией "Create debug info" ("Создать отладочную информацию"). Вы можете установить эту опцию на вкладке "Compiler" ("Компилятор") после выбора опций меню: *Options -> Customize (Опции -> Установки пользователя)* в редакторе SCL Editor. Далее выполняется компиляция при выборе команды "Create object code" ("Создать объектный код"), затем программа переносится в CPU по команде: *PLC -> Download (PLC -> Загрузить)*.

Отладчик для SCL-программ является встроенным компонентом редактора SCL Editor.

Функция "Program Status" ("Состояние программы") для SCL

С помощью этой функции отладки Вы можете отлаживать группы операторов, "ведя мониторинг области" ("monitor area") во время работы. Размер переменных "области мониторинга" зависит от используемых в программе операторов. Значения переменных в этой области обновляются и отображаются циклически.

Если область мониторинга находится в той части программы, которая выполняется в каждом программном цикле, то значения переменных из связанных в каскад циклов обычно не могут быть достоверными. Значения переменных, которые изменялись во время текущего прохода цикла, отображаются выделенным (черным) цветом шрифта, а значения, которые не изменялись, представляются светло-серым цветом.

Для того чтобы отладить SCL-программу, переключите CPU в RUN или RUN-P режим и откройте исходный файл программы. Выберите рабочий режим: *Debug -> Operation -> Debug Operation (Отладка -> Работа -> Отладка).*

Установите курсор на начало области, которую нужно отладить. Начните процесс отладки с помощью выбора опций меню: *Debug -> Monitor (Отладка -> Мониторинг)*. Имена и значения переменных в области мониторинга отображаются построчно в правой части окна редактора.

Вы можете прервать процесс отладки, выбрав вновь опции меню: *Debug -> Monitor (Отладка -> Мониторинг)*; при выборе опций меню: *Debug -> End Debug (Отладка -> Конец отладки)* процесс отладки завершается.

Функции отладки Breakpoints (точки прерывания),
Single-step Mode (пошаговый режим)

При отладке в пошаговом режиме "Single-step mode" Вы можете контролировать выполнение программы строка за строкой и при этом вести мониторинг значений переменных. Используя точки прерывания (breakpoints), Вы можете остановить программу в любой момент и отлаживать в пошаговом режиме (single-step) с этой точки программы.

Для отладки программы в пошаговом режиме (single-step) должны быть выполнены следующие требования: отлаживаемый блок не должен быть защищен, он должен быть открыт интерактивно (online) и не должен при этом модифицироваться в редакторе.

Пошаговый режим (single-step) отладки работает только на тех CPU, которые обеспечивают поддержку этого режима. Должен быть установлен режим "debug mode" (режим отладки), а отладка с использованием функции Program Status (состояние программы) должна быть выключена. CPU переходит в режим HOLD в точке прерывания (breakpoint), и отладка в пошаговом режиме (step-by-step) возможна только в режиме HOLD.

Для отладки откройте исходный файл программы и определите точки прерывания (breakpoints) установкой курсора в строке с нужным оператором и выберите опции: *Debug -> Set Breakpoint (Отладка -> Установка точки прерывания)*. Необходимо предотвратить возможность возникновения опасных состояний в установке при активации режима отладки. Для активации режима отладки выберите опции меню: *Debug -> Breakpoints Active (Отладка -> Активировать точки прерывания)*. CPU необходимо перезапустить, при достижении точки прерывания в процессе обработки программы CPU переходит в состояние паузы HOLD (см. рис. 2.12).

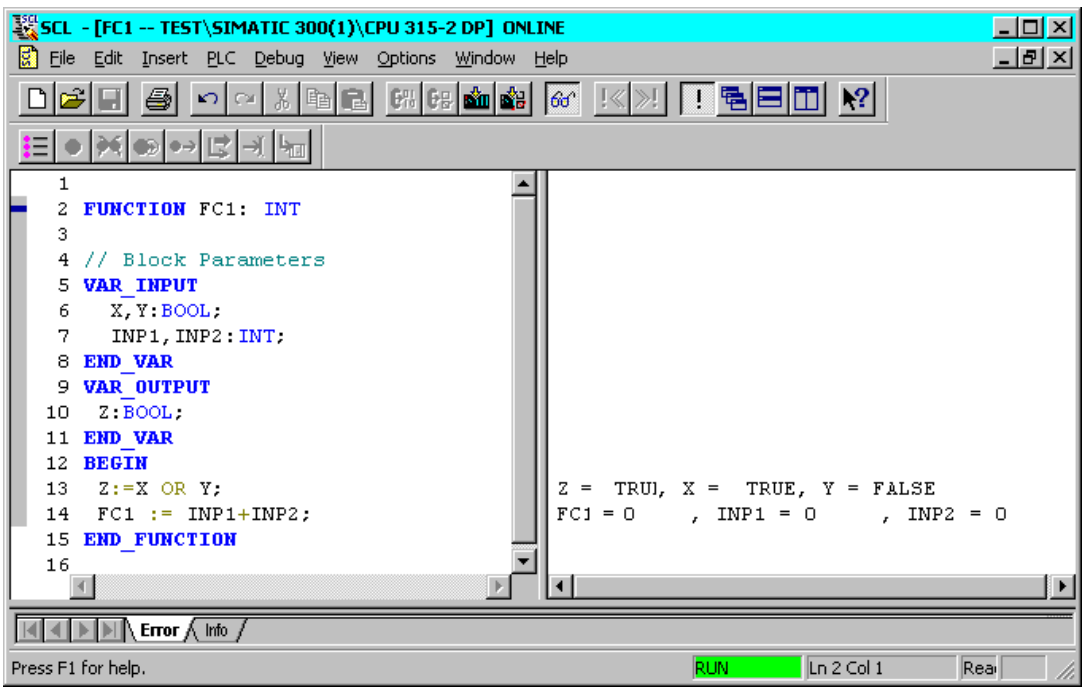


Рис. 2.12 Отладка SCL-программы

Теперь Вы можете запустить программу для выполнения в построчном (в пошаговом) режиме, выбрав опции меню: *Debug -> Execute Next Statement (Отладка -> Выполнить следующий оператор)*. Выполнение программы будет прерываться на каждом операторе, и при этом значения переменных из текущего оператора отображаются в правой части окна редактора. Отображение символьных имен может быть включено или выключено с помощью опций: *View -> Symbolic Representation (Вид -> Символьное представление)*.

С помощью опций меню: *Debug -> Resume (Отладка -> Продолжить)* программа выполняется с нормальной скоростью, пока не встретит следующую точку прерывания. При выборе опций меню: *Debug -> Execute to Selection (Отладка -> Выполнять до выбранного)* программа выполняется до раздела, выбранного с помощью курсора.

Вы можете управлять точками прерывания в программе с помощью опций меню: *Debug -> Edit Breakpoints (Отладка -> Редактировать точки прерывания)*. Вы можете также прервать процесс отладки программы с помощью повторного выбора опций меню: *Debug -> Breakpoints Active (Отладка -> Активировать точки прерывания)*.

Опции: *Debug -> End Debug (Отладка -> Завершение отладки)* вызывают завершение процесса отладки.

Примечание: с помощью опций меню: *Debug -> Execute Next Statement (Отладка -> Выполнить следующий оператор)* и *Debug -> Execute to Selection (Отладка -> Выполнять до выбранного)* устанавливаются и активируются точки прерывания. Необходимо обеспечить, чтобы при задании точек прерывания не было превышения максимально возможного количества точек прерывания, которое зависит от типа применяемого CPU.

3 SIMATIC S7-программа

В этой главе представлена структура пользовательской программы для CPU систем SIMATIC S7-300/400, начиная от различных приоритетных классов (определяющих характер выполнения программы) и отдельных частей программы (блоков) и заканчивая переменными и типами данных. В данной главе основное внимание уделяется программированию блоков на языках STL и SCL. Типы данных подробно описаны в главе 24 "Типы данных".

Вы должны определить структуру пользовательской программы на этапе разработки, когда Вы согласовываете технологические и функциональные характеристики; это будет решающим фактором при создании программы, ее отладке и тестировании. Для получения оптимальной программы необходимо обратить особое внимание на ее структуру.

3.1 Обработка программы

В целом программное обеспечение для CPU состоит из операционной системы (operating system) и пользовательской программы (user program).

Операционная система - это совокупность всех инструкций и деклараций, которые управляют системными ресурсами и процессами, использующими эти ресурсы. Операционная система включает в себя такие функции как резервирование данных в случае сбоя электропитания, активация приоритетных классов и т.п. Операционная система - это такой компонент CPU, к которому Вы, как пользователь, не имеете доступа в режиме записи. Тем не менее, Вы можете перезагружать операционную систему с модуля памяти, например, в случае обновления программы.

Пользовательская программа (user program) - это совокупность всех инструкций и деклараций (в данном случае - программных элементов), для обработки сигналов, с помощью которых установка (процесс) регулируется в соответствии с определенной задачей управления.

3.1.1 Методы обработки программы

Пользовательская программа может состоять из программных разделов, которые обрабатываются в CPU в зависимости от конкретного события. Одним из таких событий может быть, например, запуск системы автоматического управления, прерывание или обнаружение программной ошибки (см. рис. 3.1). Программы, назначенные для обработки событий, разделяются по приоритетным классам (*priority class*), с помощью которых определяется порядок обработки отдельных разделов ("mutual interruptibility" - система взаимных прерываний) программы в случаях, когда происходит одновременно несколько событий.

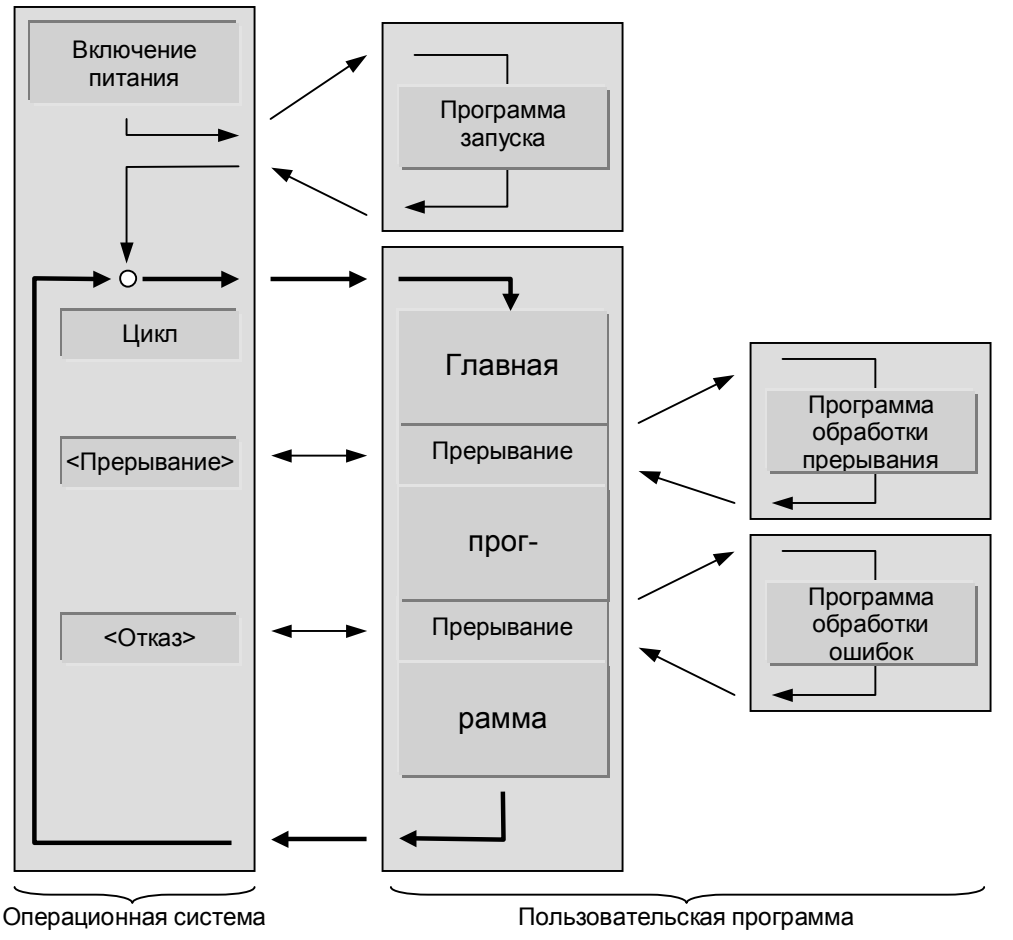


Рис. 3.1 Методы обработки пользовательской программы

Программа с наинизшим приоритетом является главной (main) программой, которая циклически обрабатывается CPU. События могут прерывать главную (main) программу в любой момент, после чего CPU выполнит связанную с прерыванием обслуживающую программу или программу обработки ошибок и затем вновь вернет управление в главную программу.

Специальный организационный блок (OB) соответствует каждому событию. Организационные блоки имеют приоритетные классы в пользовательской программе. Если происходит событие, CPU активизирует соответствующий ему организационный блок. Организационный блок (OB) - это часть пользовательской программы, которую Вы сами можете написать.

Перед тем, как CPU начнет обработку главной программы, он выполняет программу запуска (startup routine). Эта программа может быть запущена такими событиями как включение питания, поворот переключателя режимов на передней панели CPU или с помощью программатора PG. Обработка программы, следующая за выполнением программы запуска, в системе S7-300 всегда начинается с начала главной программы ("complete restart" - "полный перезапуск"), а в системе S7-400 может также использоваться режим, когда продолжается сканирование программы с точки, в которой оно было прервано ("warm restart" - "теплый перезапуск").

Главная (main) программа располагается в организационном блоке OB 1, который используется практически во всех пользовательских программах. Начало пользовательской программы соответствует первому сегменту в блоке OB 1. После завершения выполнения OB 1, что соответствует окончанию выполнения программы, CPU возвращает управление в операционную систему и после вызова различных функций операционной системы, таких как обновление отображений процесса, вновь вызывает для выполнения OB 1.

События, которые могут вмешиваться в работу программы, - это прерывания и ошибки. Источником прерываний могут стать либо процесс (аппаратное прерывание), либо CPU (прерывания по времени ["watchdog", то есть таймерные], прерывания по времени суток ["time-of-day"] и т.д.)

Что касается ошибок, то они подразделяются на синхронные и асинхронные. Асинхронные ошибки - это такие ошибки, которые не зависят от выполняемой программы, например, это может быть сбой питания на устройстве расширения или такое событие, как удаление модуля из стойки.

Синхронные ошибки - это такие ошибки, которые возникают при выполнении программы, например, ошибка будет зафиксирована при попытке доступа к несуществующему адресу или ошибка может произойти при преобразовании типов данных. Типы и номера регистрируемых событий и соответствующих организационных блоков определяются типом CPU; не каждый CPU способен обрабатывать все события, возможные в STEP 7.

3.1.2 Классы приоритетов

В таблице 3.1 показаны доступные в SIMATIC S7 организационные блоки с возможными для них классами приоритетов (приоритетными классами).

Таблица 3.1 Организационные блоки SIMATIC S7

Организационный блок	Условия вызова	Приоритет	
		по умолчанию	возможные изменения
OB свободного цикла OB 1	Периодически вызывается операционной системой	1	Нет
TOD прерывания OB 10 … OB 17	В определенное время суток или через равные промежутки времени (например, ежемесячно)	2	2 … 24
С задержкой времени OB 20 … OB 23	По истечении запрограммированного времени; управление из пользовательской программы	3 … 6	2 … 24
Watchdog прерывания OB 30 … OB 38	Регулярный вызов через запрограммированные интервалы времени (например, каждые 100 мс)	7 … 15	2 … 24
Прерывания процесса OB 40 … OB 47	Вызов по сигналу прерывания от I/O модулей	16 … 23	2 … 24
Мультипроцессорное прерывание OB 60	Вызов по приходу события в мультипроцессорном режиме; управление из пользовательской программы	25	Нет
Ошибки резервирования OB 70 OB 72 OB 73	 В случае потери резервирования из-за ошибок I/O В случае ошибки резервирования CPU В случае ошибки резервирования коммуникаций	 25 28 25	 2 … 26 2 … 28 24 … 26
Асинхронные ошибки OB 80 OB 81 … OB 84, 86, 87 OB 85	В случае ошибок, не связанных с выполнением программы (например, ошибка времени [time error], диагностическое прерывание, прерывание вставки / удаления модуля, отказ стойки / станции)	 26[2)] 26[2)] 26[2)]	 26 2 … 26 24 … 26
Фоновая обработка OB 90	Продолжительность минимального цикла еще не достигнута	29[1)]	Нет

Таблица 3.1 Организационные блоки SIMATIC S7 (*продолжение*)

Организационный блок	Условия вызова	Приоритет	
		по умолчанию	возможные изменения
Программа запуска OB 100, 101, 102	При запуске PLC	27	Нет
Синхронные ошибки OB 121, OB 122	В случае ошибок, связанных с выполнением программы (например, ошибка I/O доступа)	Приоритет OB, вызвавшего ошибку.	

[1] см. текст

[2] при запуске: 28

В некоторых классах приоритетов Вы можете изменять заданный приоритет при параметризации CPU. В таблице 3.1 для организационных блоков показаны возможные нижний и верхний приоритетные классы. Каждый CPU имеет свой диапазон значений для нижнего и верхнего приоритетных классов. В данном обзоре представлены отдельные типы CPU.

Организационный блок OB 90 (фоновая обработка) может выполняться вместо OB 1 и, как в случае с OB 1, его обработка может быть прервана любыми прерываниями и ошибками.

Программа запуска может быть в организационном блоке OB 100 (полный перезапуск) или в организационном блоке OB 101 (теплый перезапуск); она имеет приоритетный класс 27. Асинхронные ошибки, происходящие в программе запуска, имеют приоритетный класс 28. Диагностические прерывания рассматриваются как асинхронные ошибки.

Вы должны определить, какие доступные приоритетные классы Вы будете использовать при параметризации CPU. Неиспользуемые приоритетные классы (организационные блоки) должны получить приоритет 0. Соответствующие организационные блоки должны быть запрограммированы для всех используемых приоритетных классов; иначе CPU вызовет OB 85 ("Program Processing Error" - "Ошибка выполнения программы") или перейдет в режим STOP.

Для каждого выбранного приоритетного класса во области временных локальных данных (L-стек) должно быть достаточно места (более подробная информация находится в разделе 18.1.5 "Временные локальные данные").

3.1.3 Спецификации для обработки программы

Операционная система CPU обычно использует параметры, принятые по умолчанию. Вы можете изменить эти установки при параметризации CPU с помощью утилиты конфигурирования оборудования Hardware Configuration для того, чтобы параметры системы удовлетворяли Вашим особым требованиям. Вы можете изменить эти параметры в любое время.

Каждый CPU имеет свой собственный особый набор параметров. В приведенном ниже списке представлен обзор всех параметров STEP 7 и их наиболее важные установки.

- Sturtup (параметры запуска)

  Характеристика типа запуска ("cold restart" ["холодный перезапуск"] / ("warm restart" ["теплый перезапуск"]); мониторинг сигналов "Ready" или параметризации модуля; максимальная продолжительность времени, которое может произойти до момента "теплого перезапуска".

- Cycle/clock memory (цикл/тактовые меркеры)

  Включение/выключение циклического обновления отображения процесса; задание продолжительности периода мониторинга и минимальной продолжительности времени цикла; продолжительность времени цикла в процентах для коммуникаций; число тактовых меркеров; размер области отображения процесса.

- Retentive memory (реманентная память)

  Число реманентных меркеров, таймеров и счетчиков; определение реманентных областей для блоков данных.

- Memory (память)

  Максимальное количество временных локальных данных в приоритетных классах (в организационных блоках); максимальный размер L-стека и число коммуникационных заданий.

- Interrupts (прерывания)

  Спецификация приоритета аппаратных прерываний, прерываний с задержкой (time-delay interrupts), асинхронных ошибок и (возможно в скором времени) коммуникационных прерываний.

- Time-of-day Interrupts (прерывания по времени суток)

  Спецификация приоритета, спецификация стартового времени и периодичности.

- Cyclic Interrupts (циклические прерывания)

  Спецификация приоритета, спецификация времени цикла и фазового смещения.

- Diagnostics/Clock (диагностические прерывания/системные часы)

  Индикация причины перехода в состояние STOP; тип и интервал синхронизации времени; коэффициент коррекции.

- Protection (параметры доступа к программам)

  Спецификация уровня защиты, задание пароля.

- Multicomputing (параметры мультипроцессорного режима)

  Определение числа CPU.

- Integrated I/O (параметры встроенных I/O)

  Активация и параметризация встроенных I/O.

При запуске CPU загружает заданные пользователем параметры вместо параметров, принятых по умолчанию. Параметры, определенные пользователем остаются в силе до тех пор, пока они не будут заменены.

3.2 Блоки

Чтобы сделать Вашу программу более легкой для чтения и понимания, Вы можете разбить ее на такое количество частей, какое Вы пожелаете. Языки программирования STEP 7 поддерживают такой подход к программированию, обеспечивая Вас необходимыми функциями. Каждая такая часть программы должна быть самодостаточной (self-contained) и должна решать технологическую или функциональную задачу.

Рассматриваемые части программы называются "блоками" ("block"). Блок - это часть программы пользователя, характеризующаяся своими собственными функциями, структурой или функциональным назначением.

3.2.1 Типы блоков (Block Types)

Язык программирования: STL использует различные типы блоков для разных задач:

- User blocks (пользовательские блоки)

  Пользовательские блоки - это блоки, содержащие пользовательскую программу и пользовательские данные.

- System blocks (системные блоки)

  Системные блоки - это блоки, содержащие системную программу и системные данные.

- Standard blocks (стандартные блоки)

  Стандартные блоки - это готовые к использованию блоки, такие, например, как драйверы для функциональных блоков FM или коммуникационных процессоров CP.

Пользовательские блоки (User blocks)

Для больших и сложных программ "структурирование" - подразделение программы на блоки рекомендуется и отчасти является необходимостью. Вы можете выбирать среди различных типов блоков те или иные, в зависимости от условий применения.

Организационные блоки OB (Organization blocks)

Этот тип блоков служит своеобразным интерфейсом между операционной системой и пользовательской программой. Операционная система CPU вызывает организационные блоки при возникновении особого события, например, аппаратного прерывания или прерывания времени суток. Главная программа находится в организационном блоке OB 1. Остальные организационные блоки имеют постоянные назначенные номера, основанные на событиях, для обработки которых они вызываются.

Функциональные блоки FB (Function blocks)

Эти блоки являются частями программы, вызов которых может быть запрограммирован с помощью параметров блока. Они обладают областью памяти для переменных (variable memory), которая расположена в блоке данных. Этот блок данных постоянно назначен функциональному блоку, или, точнее, *вызову* функционального блока. Возможно даже назначение нескольких блоков данных (с одинаковой структурой данных, но содержащих разные значения) каждому вызову функционального блока. Такой постоянно назначенный блок данных называется *экземплярным блоком данных* (instance data block), а совокупность вызова функционального блока и экземплярного блока данных называется *экземпляром вызова* (call instance) или, для краткости, "экземпляром" ("instance"). Функциональные блоки могут также хранить свои переменные в экземплярном блоке данных вызывающего функционального блока; тогда такой экземплярный блок данных называется "локальным экземпляром" ("local instance").

Функции FC (Functions)

Функции используются для программирования часто повторяющихся или сложных функций автоматики (automation functions). Функциям могут назначаться параметры. Функции могут возвращать значение (значение вызванной функции) в вызывающий блок. Причем значение функции - необязательный параметр. Кроме функционального значения функция может иметь другие выходные параметры. Функции не сохраняют информацию и не имеют назначенных блоков данных.

Блоки данных DB (Data blocks)

Эти блоки содержат данные Вашей программы. Программируя блоки данных, Вы определяете, в какой форме данные будут сохраняться (в котором блоке, в каком порядке и с каким типом данных). Существует два способа использования блоков данных: как блоки глобальных данных (global data blocks) и как экземплярные блоки данных (instance data blocks). Блоки глобальных данных в пользовательской программе являются, как говорится, "свободными" ("free") блоками данных и не назначаются кодовому блоку. Экземплярные блоки данных, однако, назначаются функциональному блоку и сохраняют часть локальных данных этого функционального блока.

Максимальное число для каждого типа блоков и размер этих блоков определяются типом CPU. Число организационных блоков и их номера фиксированы; они назначаются операционной системой CPU. Блокам других типов Вы можете самостоятельно назначать номера внутри определенных пределов. Также Вы можете выбрать для каждого блока имя (символ) в таблице символов, с тем, чтобы ссылаться на блок по символьному имени.

Системные блоки (System blocks)

Системные блоки (System blocks) являются компонентами операционной системы. Они могут содержать программы (системные функции SFC или системные функциональные блоки SFB) или данные (системные блоки данных SDB). Системные блоки предоставляют множество важных системных функций, доступных пользователю, таких, например, как функции управления внутренними часами CPU или различные коммуникационные функции.

Вы можете вызывать SFC и SFB, но Вы не можете ни изменить эти функции, ни запрограммировать их самостоятельно. Собственно системные блоки не занимают места в пользовательской памяти (user memory); только вызовы блоков и экземплярные блоки данных для SFB располагаются в пользовательской памяти.

Системные блоки данных SDB содержат информацию о таких вещах, как конфигурация автоматизированной системы или параметры модулей. Система STEP 7 самостоятельно генерирует эти блоки и управляет ими.

Тем не менее, Вы можете определять их содержимое, например, когда Вы конфигурируете станции. Как правило, системные блоки данных SDB размещаются в загрузочной (load) памяти. Пользователь не имеет доступа к ним из своей программы.

Стандартные блоки (Standard blocks)

В дополнение к функциям и функциональным блокам, которые Вы можете создавать самостоятельно, Вы можете использовать готовые для применения блоки, так называемые "стандартные блоки" ("Standard blocks").

Эти блоки могут быть получены или на различных носителях, или могут быть в составе библиотек из комплекта поставки ПО STEP 7 (например, IEC-функции или функции для S5/S7 конвертирования).

В главе 33 "Библиотеки блоков" представлен обзор стандартных блоков из состава "*библиотеки стандартных блоков*" *Standard Library*.

3.2.2 Структура блоков (Block Structure)

На нижеприведенных рисунках представлены структуры блоков для случаев "инкрементного" программирования и программирования, ориентированного на создание исходных текстов программы:

Логический блок (logic block) ("инкрементное" программирование)

Block header (заголовок блока)

Declaration (раздел объявления)

Address (адрес)	*Declaration (объявление)*	*Name (имя)*	*Type (тип)*	

Program (программа)

```
A  Input1      //Limit switch responded      (отклик конечного выключателя)
A  Input2      //Manual operation            (ручной режим)
=  Output1     //Message to operator panel (сообщение на панель оператора)
```

Блок данных (data block) ("инкрементное" программирование)

Block header (заголовок блока)

Declaration (раздел объявления)

Address (адрес)	*Name (имя)*	*Type (тип)*	*Initial Value (начальное значение)*	

Рис. 3.2 Структура блока ("инкрементное" программирование)

Логический блок (logic block) (программирование, ориентированное на создание исходных текстов программы)

```
Block type Address (Тип блока Адрес)
Block header (заголовок блока)

VAR_xxx

name : Data type := Initialization;
name : Data type := Initialization;
…

END_VAR

BEGIN

Program          (программа)

END_Block Type
```

Блок данных (data block) (программирование, ориентированное на создание исходных текстов программы)

```
DATA_BLOCK Address (Блок данных Адрес)
Block header (заголовок блока)

STRUCT

name : Data type := Initialization;
name : Data type := Initialization;
…

END_ STRUCT

BEGIN

name := Initialization;

END_DATA_BLOCK
```

Рис. 3.2 Структура блока (программирование, ориентированное на создание исходных текстов программы)

По существу кодовые блоки состоят из трех частей:

- Block header (заголовок блока), который содержит свойства (характеристики) блока, такие как имя блока.
- Declaration section (раздел объявления), в котором декларируются (т.е. определяются) локальные ("block-local" - "внутриблочные") переменные данного блока.
- Program section (раздел программы), который содержит саму программу и комментарии к ней.

Блоки данных имеют похожую структуру:

- Block header (заголовок блока), который содержит описание свойств (характеристик) блока.
- Declaration section (раздел объявления), в котором объявляются локальные ("внутриблочные") переменные; в этом случае с адресами данных указываются типы данных.
- Initialization section (раздел инициализации), в котором отдельным адресам данных назначаются начальные значения.

В случае "инкрементного" программирования раздел объявления переменных и раздел инициализации объединены. Вы определяете адреса данных, их типы данных в "declaration view" (вид "объявлений") и также Вы можете инициализировать каждый адрес данных отдельно в "data view" (вид "данных") (см. ниже).

3.2.3 Свойства блоков (Block Properties)

Свойства блоков или атрибуты содержатся в заголовке блока. Вы можете увидеть и изменить атрибуты блока в редакторе с помощью опций меню: *File -> Properties (Файл -> Свойства)* (см. рис. 3.3).

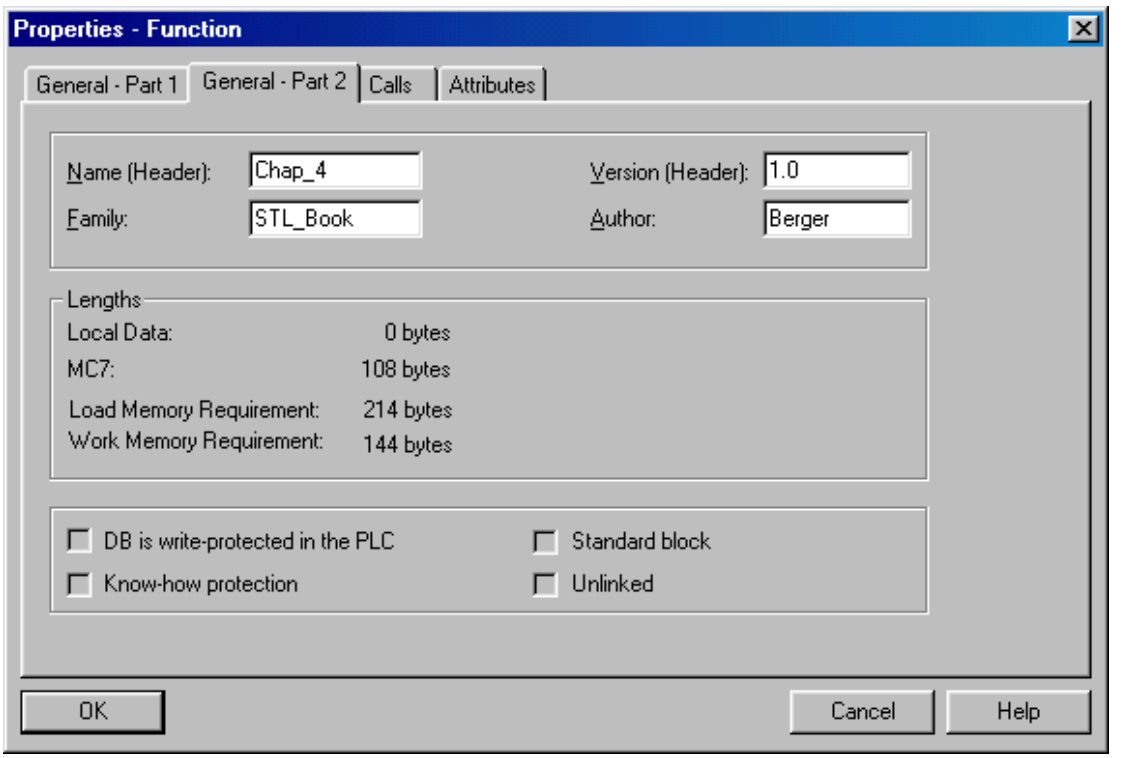


Рис. 3.3 Окно свойств блока ("Properties - *Type* block")

На вкладке "General - Part 2" ("Общие - часть 2") показано распределение памяти для блока в байтах:

- Local Data (временные локальные данные - размещение стека локальных данных).
- MC 7: размер блока (только код).
- Load memory requirement (требования к загрузочной памяти).
- Work memory requirement (требования к рабочей памяти).

Атрибут "*Know-how protection" ("защита технологии")* используется для защиты блока. Если блок защищен с помощью установки этого атрибута, то программа из этого блока не может быть отображена, распечатана или изменена. В редакторе можно будет увидеть только заголовок блока и таблицу объявления переменных (declaration table) с параметрами блока. При вводе текста в исходный файл Вы можете защитить каждый блок сами с помощью ключевого слова KNOW_HOW_PROTECT. Если Вы защитили блок, то никто (даже Вы) не сможет увидеть компилированный вариант этого блока (Вы должны обеспечить при этом безопасное место хранения для исходного файла программы!).

Заголовок блока любого стандартного блока (*standard block*), который предоставляется фирмой Siemens, содержит атрибут "S*tandard Block*".

Атрибут "*DB is write-protected in the PLC" ("DB в PLC доступен только для чтения")* используется только для блоков данных. Установка этого атрибута приведет к тому, что Вы сможете только считывать данные в Вашей программе. При попытке записи в защищенный блок данных выводится сообщение об ошибке. Такой вариант защиты (защита от записи) нельзя путать с защитой блока. Блок данных с защитой блока может быть считан и перезаписан в пользовательской программе, но его данные нельзя отобразить ни с помощью программатора PG, ни с помощью устройств наблюдения оператора.

Блок данных с установленным атрибутом "*Unlinked*" ("*Неподключенный*") будет находиться только в загрузочной (load) памяти; он не может быть запущен на выполнение ("non execution-relevant"). Вы не сможете записывать в блоки данных, находящиеся в загрузочной памяти, и Вы сможете считать данные этих блоков только с помощью системной функции SFC 20 BLKMOV.

Прочие спецификации вкладки "General - Part 2" ("Общие - часть 2") окна свойств блока:

Атрибут *Name (Имя)* идентифицирует блок; это не то же самое, что и символьный адрес: разные блоки могут иметь одинаковое имя.

Атрибут *Family (Семья - имя группы - второе имя)* позволит Вам назначить общие характеристики для группы блоков. Идентификаторы блока *Name (Имя)* и *Family (Семья - имя группы)* отображаются при вставке блоков и при выборе блоков в диалоговом окне каталога элементов программы (program elements catalog).

Атрибут *Author (Автор)* идентифицирует создателя блока.

Атрибуты *Name (Имя), Family (Семья - имя группы), Author (Автор)* могут содержать до 8 символов (здесь могут применяться следующие символы: буквы, цифры и знак подчеркивания).

Атрибут *Version (Версия)* вводится дважды двумя цифрами: от 0 до 15.

На вкладке "General - Part 1" ("Общие - часть 1") редактор записывает дату изменения блока в две отметки времени: для кодового блока и для интерфейса, т.е. для блока параметров и для статических локальных данных.

Примечание: надо отметить, что дата изменения интерфейса должна быть такой же или более ранней, нежели дата изменения программного кода вызывающего блока. Если это условие не выполняется, то при выводе на экран вызывающего блока редактор сигнализирует об ошибке "time stamp conflict" ("конфликт отметок времени").

Блоки могут быть созданы или скомпилированы в виде версии 1 или в виде версии 2. Это имеет практическое значение только для функциональных блоков. Если активировано свойство "multi-instance capability" ("несколько экземпляров DB для функционального блока"), что, кстати, является обычным случаем, то мы имеем дело с блоком версии 2. Если свойство "multi-instance capability" выключено, то Вы не сможете вызвать этот блок как локальный экземпляр, также как Вы не сможете вызвать другой функциональный блок из этого блока как локальный экземпляр. Функциональный блок версии 1 имеет преимущество, заключающееся в ограничении использования экземпляра блока данных в случае косвенной адресации (имеет значение только при STL-программировании).

На вкладке "Calls" ("Вызовы") Вы увидите список всех блоков, вызываемых в данном блоке с отметками времени для кодовых блоков и интерфейса.

На вкладке "Attributes" ("Атрибуты") показаны системные атрибуты блока. С помощью системных атрибутов осуществляется координация и управление функциями разных приложений, например, в системе управления SIMATIC PCS7.

Program length (Размер программы)

Длина пользовательской программы содержится в свойствах (Properties) в автономном каталоге *Blocks (Блоки).* Для доступа к этим данным выберите *Blocks (Блоки)* и используйте опции: *Edit -> Object Properties (Правка -> Свойства объекта).* Теперь на вкладке "Blocks" ("Блоки") Вам доступна информация "Size in work memory" (Размер в рабочей памяти) и "Size in load memory" (Размер в загрузочной памяти).

Примечание: примите во внимание, что конфигурационные данные (системные блоки данных) не учитываются при указании размера программы в загрузочной (load) памяти. Открыв каталог *Blocks (Блоки),* Вы можете увидеть требования к загрузочной (load) памяти для системных данных в деталях (представленных в табличной форме). В строке состояния утилита SIMATIC Manager указывает суммарный объем памяти для всех блоков, которые Вы выберете с нажатой клавишей Ctrl.

С помощью программатора PG, подключенного интерактивно (online), при использовании утилиты SIMATIC Manager Вы можете найти текущие назначения памяти CPU на вкладке "Memory" ("Память"), используя опции меню: *PLC -> Module Information (PLC -> Информация о модуле).*

Контрольная сумма (Checksum)

Редактор программы Program Editor генерирует контрольную сумму (Checksum) для всех блоков пользовательской программы и сохраняет ее в свойствах объекта каталога *Blocks (Блоки).* Идентичные программы имеют одинаковую контрольную сумму, контрольная сумма изменяется при любом изменении программы.

Контрольная сумма также генерируется для системных данных. Для доступа к контрольным суммам при помощи утилиты SIMATIC Manager выберите каталог Blocks (Блоки) и используйте опции: *Edit -> Object Properties (Правка -> Свойства объекта).*

3.2.4 Интерфейс блоков (Block Interface)

Таблица объявления переменных содержит интерфейс блока с остальной программой. Он состоит из параметров блока (входы, выходы и входные и выходные параметры), а также статических локальных данных (для функциональных блоков). Временные локальные данные не принадлежат интерфейсу блока. Интерфейс блока определяется в таблице объявления переменных, и эти переменные инициализируются при вызове блока (см. главу 19. "Параметры блоков").

Редактор программ Program Editor проверяет, чтобы инициализация параметров вызываемого блока соответствовала интерфейсу вызываемого блока. Для этого редактор использует метки времени: интерфейс вызываемого блока должен иметь более раннюю временную метку, чем код вызывающего блока, что означает, что последние изменения интерфейса должны быть выполнены раньше его объединения с блоком. Редактор программ Program Editor обновляет метку времени интерфейса при изменении числа параметров, или при изменении типа данных, или при изменении значений параметров, принимаемых по умолчанию.

Конфликт временных меток (Time stamp conflict)

Если интерфейс вызываемого блока имеет более позднюю временную метку, чем код вызывающего блока, возникает "конфликт временных меток" ("Time stamp conflict"). Так, Вы получите "конфликт временных меток" ("Time stamp conflict"), если вновь откроете уже скомпилированный блок. В этом случае редактор Program Editor выделит некорректный вызов блока красным цветом. Конфликт временных меток также возникнет, если Вы, например, измените интерфейсы блоков, которые уже вызывались в других блоках, или если Вы объедините блоки из разных программ в новую программу, или если Вы перекомпилируете раздел полной программы из исходного файла.

Тем не менее, конфликт интерфейса, в общем описываемый как "конфликт временных меток" ("Time stamp conflict"), может также иметь другие причины. Он может случиться, если вызванный или адресованный (referenced) блок имеет более позднюю временную метку (younger), чем вызывающий блок. Ниже представлены примеры возможных случаев "конфликта временных меток" ("Time stamp conflict"):

- Интерфейс вызываемого блока имеет более позднюю временную метку (younger), чем код вызываемого блока.
- Интерфейс инициализации не согласован с интерфейсом блока.
- Функциональный блок имеет более позднюю временную метку (younger), чем его экземплярный блок данных (экземпляр DB генерируется на основе описания интерфейса функционального блока и должен, следовательно, иметь более позднюю временную метку, чем метка функционального блока, или их метки должны быть синхронны).
- Интерфейс локального экземпляра имеет более позднюю временную метку, чем вызывающий экземпляр (касается функциональных блоков).

- Пользовательский тип данных UDT имеет более позднюю временную метку (younger), чем блок, переменные которого объявлены как UDT; это может быть любой блок, включая блок данных или другой UDT.

Корректировка неправильных вызовов блока

Редактор программ Program Editor обеспечивает возможность исправления некорректных вызовов блока или UDT-приложений при выборе команд меню: *Edit -> Block Call -> Update (Правка -> Вызов блока -> Обновить).* Для случая одинаковых имен, типов данных или местоположения редактор может найти правильные назначения в большинстве случаев. Если этого не произошло, то Вы должны выполнить корректировку вручную. В любом случае Вы должны проверить правильность выполненной корректировки.

Check Block Consistency (Проверка блока на консистентность)

Редактор программ Program Editor лишь информирует о наличии "конфликта временных меток" ("Time stamp conflict"), если Вы открываете блок, содержащий этот самый "конфликт временных меток". Если необходимо проверить программу целиком, Вы можете использовать функцию проверки консистентности блока "Check Block Consistency". Эта функция снимает большинство конфликтов интерфейса и указывает на места в программе, требующие редактирования.

Для выполнения проверки на консистентность данных выберите каталог *Blocks (Блоки)* и затем опции меню: *Edit -> Check Block Consistency (Правка -> Проверки консистентности блока).* Редактор программ Program Editor генерирует данные, требующиеся для такой проверки, начиная с системы STEP 7 V5.0 SP3. Если пользовательская программа скомпилирована в системе STEP 7 более ранней версии или если программа содержит блоки, скомпилированные в системе STEP 7 более ранней версии (Вам необходимо будет проверить это, если соответствующая информация не отображается в окне функции "Check Block Consistency"), выбрав в этом окне: *Program -> Compile (Программа -> Компилирование)*.

Редактор программ Program Editor отобразит процесс выполнения задачи и результат проверки на консистентность в окне результата ("1:Compile"). Такая проверка консистентности не может быть использована для программ, находящихся в библиотеках.

Отношения в случае вызванных или адресованных блоков отображаются в форме древовидной диаграммы (рис. 3.4).

Вы можете выбирать между двумя представлениями, указанными ниже.

"Дерево ссылок" (*reference tree*) представляет связи аналогично отображению структуры программы: слева расположены вызывающие блоки, а правее расположены вызванные ими блоки. Пример: экземпляр DB 20 / FB 20 вызван в OB 1, а локальные экземпляры FB 21 и FB 22 вызваны в FB 20.

"Дерево подчиненности" (*dependency tree*) представляет связи, начиная от всех вызванных или адресованных блоков. Эти блоки расположены в левом ряду, а правее расположены вызывающие их блоки. Пример: FB 22 хранит свои данные в экземпляре DB 20 / FB 20, который вызывается в OB 1. Он также имеет свой собственный DB 29, и он вызывается как локальный экземпляр в FB 20.

Для обоих представлений случай появления знака восклицания (!) будет означать необходимость исправления и компиляции соответствующего блока.

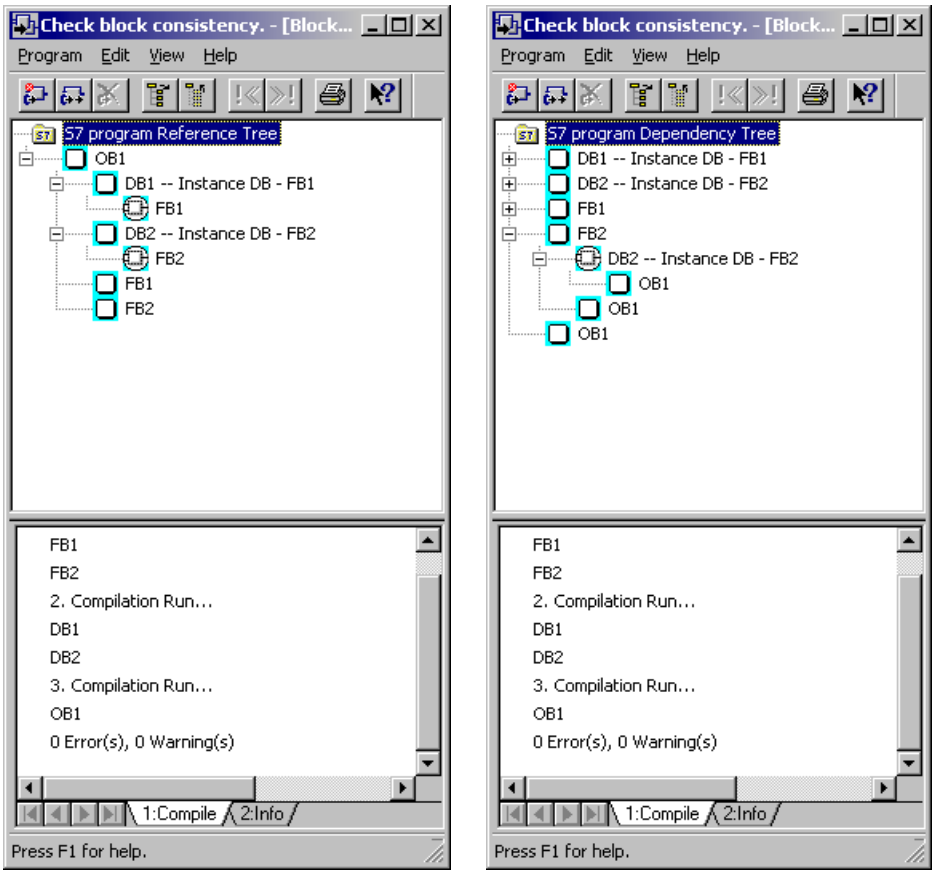


Рис. 3.4 Пример представления структуры отношений блоков по результатам проверки консистентности данных с помощью функции Check Block Consistency

Если Вы выбрали блок в древовидной схеме или в открытом окне, Вы сможете отредактировать его, выбрав соответствующие опции меню: *Edit -> Open Block (Правка -> Открыть блок)*, т.е. Вы можете исправить некорректный вызов.

3.3 Адресация переменных (Addressing Variables)

При адресации переменных Вы можете выбирать способ адресации из двух основных вариантов: абсолютная адресация (absolute addressing) или символьная адресация (symbol addressing). При абсолютной адресации используются численные адреса, начиная с нулевого (0) адреса для каждой адресной области. При символьной адресации используются символьные (состоящие из букв и цифр) имена, которые Вы сами задаете в таблице символов (Symbol Table) для глобальных адресов или в разделе объявления переменных (declaration section) для внутриблочной адресации. Расширением абсолютной адресации является косвенная адресация (indirect addressing), при которой адреса (местоположение) в памяти высчитываются во время выполнения программы.

3.3.1 Абсолютная адресация переменных

Переменные простых типов могут быть адресованы с использованием абсолютной адресации (absolute addressing).

Абсолютные адреса входов и выходов рассчитываются, исходя из начального адреса модуля, который Вы установили или должны установить в таблице конфигурации (configuration table), и типа сигнала, подключаемого к модулю. Подключаться могут как дискретные, так и аналоговые сигналы.

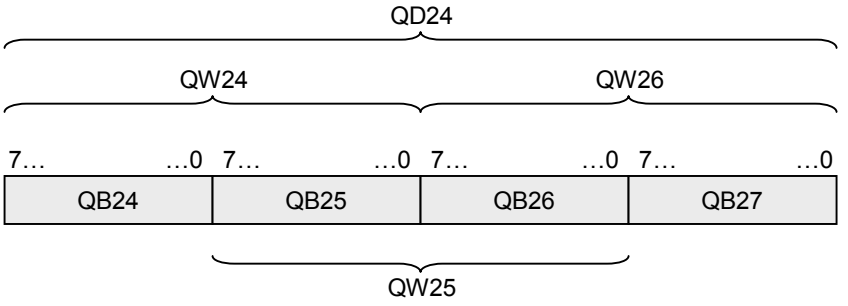


Рис. 3.5 Биты и байты в машинных словах и в двойном слове

Дискретные (binary) сигналы

Дискретный сигнал содержит один бит информации. Примерами дискретных сигналов являются входные сигналы от конечных выключателей, кнопок и т.п., которые поступают на дискретные входные модули, и выходные сигналы, управляющие лампами, контакторами и т.п., которые поступают на дискретные выходные модули.

Аналоговые сигналы

Аналоговый сигнал содержит 16 бит информации. Аналоговый сигнал соответствует "каналу" ("channel"), который занимает в контроллере машинное слово (word), т.е. 2 байта (см. ниже). Аналоговые входные сигналы (например, напряжения от терморезисторов) поступают на аналоговые входные модули, оцифровываются и после этого становятся доступными для обработки в контроллере в виде 16-разрядного сигнала (16 информационных битов). С другой стороны, 16-разрядный сигнал может управлять аналоговым индикатором посредством преобразования в аналоговом выходном модуле в аналоговый сигнал (например, ток). Динамическому диапазону изменения ("information width" - "информационный диапазон") сигнала соответствует динамический диапазон изменения ("information width") переменной, в виде которой сигнал сохраняется и обрабатывается. Динамический диапазон изменения сигнала и интерпретация этого сигнала (например, относительное положение), взятые вместе, определяют тип данных (*data tape*) для соответствующей переменной.

Дискретные сигналы сохраняются в переменных типа BOOL (булева переменная), аналоговые сигналы - в переменных типа INT (целая переменная).

Определяющим фактором для адресации переменной является ее тип, от которого зависит требуемая величина области памяти для размещения переменной.

В системе STEP 7 существуют 4 типа данных для абсолютной адресации:

- 1 бит Тип данных BOOL,
- 8 битов Тип данных BYTE или другой 8-битовый тип данных,
- 16 битов Тип данных WORD или другой 16-битовый тип данных,
- 32 бита Тип данных DWORD или другой 32-битовый тип данных.

На переменные типа BOOL ссылка производится посредством идентификатора адреса, номера байта и отделенного десятичной точкой номера бита. Нумерация байтов начинается с нуля (0) в каждой адресной области. Верхнее предельное значение номера байта определяется типом CPU. Биты внутри байтов нумеруются от 0 до 7.

Примеры:

I 1.0 входной бит с номером 0 в байте номер1

Q 16.4 выходной бит с номером 4 в байте номер16

Для переменных типа BYTE в качестве абсолютного адреса используется идентификатор адреса и номер байта, в котором содержится собственно значение переменной. Идентификатор адреса дополнен символом B.

Примеры:

IB 2 входной байт номер 2

QB 18 выходной байт номер 18

Переменные типа WORD состоят из двух байтов (слово). В качестве абсолютного адреса используется идентификатор адреса и номер младшего байта машинного слова, в котором содержится собственно значение переменной. Идентификатор адреса дополнен символом W.

Примеры:

IW 4 входное слово номер 4; содержит байты 4 и 5

QW 20 выходное слово номер 20; содержит байты 20 и 21

Переменные типа DWORD состоят из четырех байтов (двойное слово). В качестве абсолютного адреса используется идентификатор адреса и номер младшего байта двойного слова, в котором содержится собственно значение переменной. Идентификатор адреса дополнен символом D.

Примеры:

ID 8 входное двойное слово номер 8; содержит байты 8, 9, 10 и 11

QD 24 выходное двойное слово номер 24; содержит байты 24, 25, 26 и 27

Адресация области данных в блоке данных.

Примеры:

DB 10.DBX 2.0 бит данных 2.0 в блоке данных DB 10

DB 11.DBB 14 байт данных 14 в блоке данных DB 11

DB 20.DBW 20 слово данных 20 в блоке данных DB 20

DB 22.DBD 10 двойное слово данных 10 в блоке данных DB 22

Дополнительную информацию по адресации областей данных Вы найдете в разделе 18.2.2 "Адресация данных".

3.3.2 Косвенная адресация

Косвенная адресация (indirect addressing) позволяет рассчитывать адреса в области данных во время выполнения программы. Языки программирования STL и SCL используют различные методы для косвенной адресации. В STL различают следующие виды адресации:

- Косвенная адресация посредством памяти
  ("Memory-indirect-addressing")
  Например, IW [MD 200]
  означает, что адрес находится двойном слове памяти.
- Косвенная внутризонная адресация посредством регистра
  ("Register-indirect area-internal addressing")
  Например, IW [AR1, P#2,0]
  означает, что адрес, находящийся в адресном регистре AR1, получает приращение на величину смещения (offset) P#2,0 при выполнении оператора.
- Косвенная межзонная адресация посредством регистра
  ("Register-indirect area-crossing addressing")
  Например, W [AR1, P#0,0]
  означает, что адрес (включая адресную область), находящийся в адресном регистре AR1, получает приращение на величину смещения (offset) P#0,0 при выполнении оператора.

Двойные слова адресной области для данных (DBD и DID), меркеров (MD) и временных локальных данных (LD) могут использоваться для хранения адресов при косвенной адресации посредством памяти. Косвенную адресацию посредством регистра можно применять с использованием двух адресных регистров: AR1 и AR2.

Косвенная адресация подробно описана в главе 25 "Косвенная адресация".

При использовании языка программирования SCL адресные области состоят из поля, элементы которого доступны косвенно и отдельно. Например, MW[*index*] - это обращение к слову памяти, адрес которого размещен в переменной *index.* Переменная *index* может быть изменена в процессе выполнения программы. Более подробно косвенная адресация при использовании SCL рассматривается в разделе 27.2.3 "Косвенная адресация при использовании SCL".

3.3.3 Символьная адресация переменных

Символьная адресация (symbolic addressing) использует имена (символы) вместо абсолютных адресов. Вы сами можете выбирать эти имена. Такое имя должно начинаться с буквы и может содержать до 24 символов. В STL не разрешено использовать ключевые слова в качестве имен (символов). Для того, чтобы использовать ключевые слова в качестве имен (символов) в SCL, вставьте символ решетки "#" перед таким именем.

При присвоении имен входам учитывается регистр написания символа (имеет значение, какой регистр применяется: верхний или нижний). Для имен выходов редактор использует регистр и нотацию (форму записи), которые были применены при объявлении (declaration) символа.

При символьной адресации абсолютному адресу должно быть назначено имя (символ).

Символы различаются по месту назначения: глобальные символы действительны во всей программе, тогда как локальные символы действительны только в блоке, в разделе объявления переменных которого они описаны.

Глобальные символы

Вы можете назначить имена в таблице символов (symbol table) следующим объектам:

- Блоки данных и кодовые блоки
- Входы, выходы, периферийные входы и периферийные выходы
- Меркеры, таймеры и счетчики
- Пользовательские типы данных
- Таблицы переменных

Глобальный символ может также содержать пробелы, специальные символы и национальные символы, такие как умляут. Исключения составляют символы 00_{hex}, FF_{hex} и кавычки ("). При использовании в программе имен со специальными символами Вы должны заключать имена в кавычки. В скомпилированном блоке программный редактор всегда отображает все глобальные символы в кавычках.

Вы можете использовать глобальный символ во всей программе; каждый такой символ должен быть уникальным (однозначно принадлежать одному адресу) в этой программе.

Редактирование, импортирование и экспортирование глобальных символов описано в разделе 2.5.2 "Таблица символов".

Локальные символы

Имена локальных данных определяются в разделе объявления переменных соответствующего блока. Эти имена могут содержать только буквы, цифры и знак подчеркивания.

Локальные символы являются действующими только внутри блока, в котором они описаны. Такие же символы (такие же имена переменных) могут быть применены в ином контексте (для обозначения совершенно иных объектов) в другом блоке. Редактор отображает локальные символы (имена) с впереди стоящим символом "#". Если редактор не может отличить локальный символ от адреса Вы должны вводить этот символ с впереди стоящим символом "#".

Локальные символы доступны только в базе данных программатора PG (в автономном [offline] каталоге *Blocks* [*Блоки*]). Если эта информация отсутствует при декомпиляции, то редактор вставляет символы замены (substitute symbol).

Использование символьных имен

Если Вы используете символьные имена во время инкрементного программирования, то эти имена должны уже к этому времени быть присвоены абсолютным адресам. Вы можете ввести новые символические имена в таблицу символов во время инкрементного программирования и можете в дальнейшем использовать в программе.

Если вы программируете исходный текстовый файл программы, то полностью закончить процесс назначения символических имен абсолютным адресам необходимо лишь к моменту компиляции программы.

В случае использования массивов доступ к отдельным элементам массивов обеспечивается использованием имени массива с индексом, например, имя MSERIES[1] принадлежит первому элементу массива MSERIES. В случае программирования на STL индекс должен быть константой (INT). В случае программирования на SCL индекс может быть как целой переменной (INT), так и целым выражением (INT).

В структурах каждый элемент имени ("subname") отделяется от остальных элементов десятичной точкой, например, FRAME.HEADER.CNUM.

Компоненты пользовательских типов данных адресуются точно также как и компоненты структур.

Подробная информация изложена в главе 24 "Типы данных".

Адресация данных

Символьная адресация данных предполагает использование полного адреса. включая блока данных. Например, блок данных с символьным адресом MVALUES содержит переменные MVALUE1, MVALUE2 и MTIME. Эти переменные могут быть адресованы следующим образом:

"MVALUES". MVALUE1

"MVALUES". MVALUE2

"MVALUES". MTIME

Для получения дополнительной информации по использованию адресов для доступа к данным обратитесь к разделам 18.2.2 "Адреса для доступа к данным" (STL) и 27.2.2 "Символьная адресация" (SCL).

3.4 Программирование кодовых блоков на STL

3.4.1 Структура STL-выражения

STL-программа состоит из ряда отдельных выражений (statement). Выражение - это наименьшая самостоятельная единица пользовательской программы. Выражение содержит описание работы для CPU. На рисунке 3.6 показана общая структура STL-выражения.

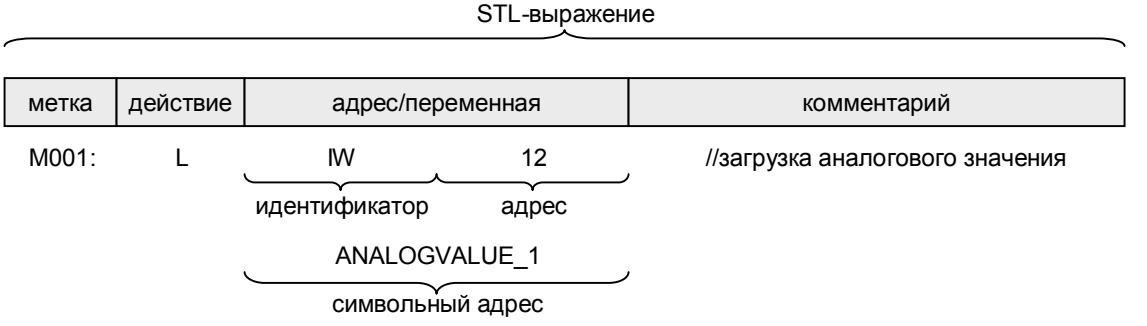


Рис. 3.6 Структура STL-выражения

Ниже перечислены компоненты STL-выражения:

- Метка (не обязательный элемент) содержит до 4 символов, заканчивается двоеточием ":" (см. раздел "Функции перехода").
- Описание задания для CPU (такие задания, например, как load [загрузить], scan [считать], compare [сравнить] и т.д.).
- Адрес - информация, необходимая для выполнения действия (например, абсолютный адрес IW12, символьный адрес некоторой переменной ANALOGVALUE_1 или некоторой константы W#16#F001 и т. д.). Отдельные операторы не требуют задания адреса.
- Комментарий (не обязательный элемент) должен начинаться двумя косыми чертами "//" и может продолжаться до конца строки.

При вводе в исходный файл Вы должны заканчивать каждое выражение (до начала комментария, если он есть) символом "точка с запятой (";"). В STL строка может содержать не более 200 символов, а длина комментария не может быть больше 160 символов.

3.4.2 Инкрементное программирование кодовых блоков на STL

В разделе 2.5 "Создание S7-программ" Вы можете найти введение в основы создания S7-программы и использование программного редактора.

Создание блоков

Процесс программирования блока начинается с его открытия одним из двух способов: либо двойным щелчком на блоке в окне проекта SIMATIC Manager, либо в редакторе с помощью выбора опций меню: *File -> Open (Файл -> Открыть).* Если блок пока не существует, Вы должны его сначала создать одним из следующих путей:

- В левой половине окна проекта SIMATIC Manager выберите объект *Blocks (Блоки),* создайте новый блок с помощью опций меню: *Insert->S7 Block->… (Вставка -> S7 Block -> …)*. В окне свойств (Properties) блока на вкладке "General - Part 1" ("Общие - часть 1") выберите номер блока и язык программирования "STL".
- Находясь в редакторе, с помощью опций меню: *File -> New (Файл -> Создать)* вызовите окно диалога с заголовком блока (номер блока, язык программирования, атрибуты блока). После закрытия диалогового окна Вы можете вводить программу этого блока.

Вы можете ввести информацию заголовка блока либо при создании блока, либо позднее, активизировав редактор, затем открыв блок и выбрав опции меню: *File -> Properties (Файл -> Свойства)*.

В редакторе программ язык программирования устанавливается на вкладке "Create Block" ("Создать блок") в диалоговом окне, открытом с помощью опций меню: *Options -> Customize (Опции -> Установки пользователя)*.

Окно блока

На нижеприведенном рисунке представлен пример открытого блока STL-программы (см. рис. 3.7).

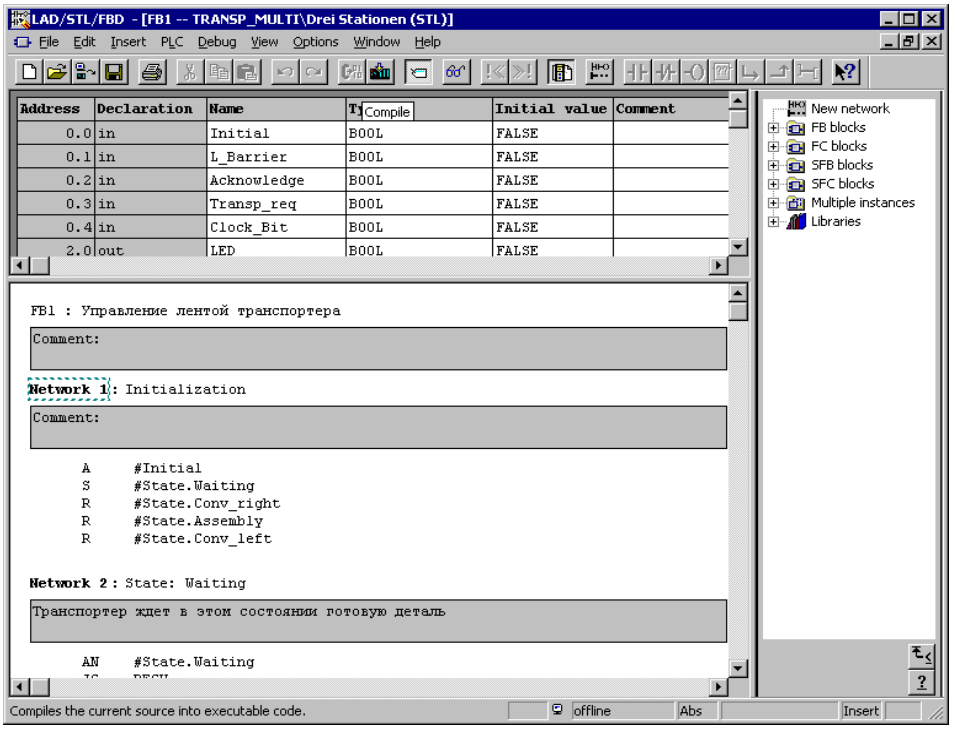


Рис.3.7 Пример открытого STL-блока

Когда открывается кодовый блок, появляется окно блока, которое само по себе состоит из трех частей:

- В верхней части окна блока отображается окно таблицы объявления переменных. Именно здесь Вы определяете внутриблочные (локальные) переменные данного блока.
- Ниже таблицы объявления переменных отображается окно программы. В этом окне Вы вводите текст Вашей программы для блока.
- В правой части окна блока отображается окно каталога элементов программы. В STL этот каталог содержит только блоки, которые находятся в автономном (offline) каталоге *Blocks (Блоки)*, а также уже запрограммированные экземпляры мультиэкземплярных FB и доступные библиотеки.

Таблица объявления переменных

Таблица объявления переменных располагается выше окна программы. Если она не видна, то поместите курсор на верхней разделительной с окном программы линии, щелкните левой кнопкой мыши, когда курсор изменит свою форму, и "потащите" курсор. Вы увидите окно таблицы объявления переменных, в которой Вы должны дать описание для локальных переменных блока (см. таблицу 3.2). Существуют типы переменных, которые не могут использоваться в некоторых типах кодовых блоков. Если Вы не используете данный тип переменных, соответствующая строка должна остаться незаполненной.

Таблица 3.2 Типы переменных в разделе объявления переменных

Variable type (Тип переменной)	Declaration (Описание)	Тип переменной возможен в блоках типов		
Input parameters (Входные параметры)	in	-	FC	FB
Output parameters (Выходные параметры)	out	-	FC	FB
In-out parameters (Вх/вых параметры)	in_out	-	FC	FB
Static local data (Статические локальные данные)	stat	-	-	FB
Temporary local data (Временные локальные данные)	temp	OB	FC	FB

Описание переменных состоит из имени, типа данных, значения по умолчанию (если есть) и комментария (не обязательный элемент). Не всем переменным может быть назначено значение по умолчанию (например, значения по умолчанию не могут присваиваться временным локальным данным). Более подробно значения, назначаемые по умолчанию для функций и функциональных блоков, описаны в главе 19 "Параметры блока".

Порядок описаний в кодовом блоке фиксирован (как показано в таблице выше), в то же время порядок следования типов переменных произволен. Вы можете экономно расходовать память, если будете группировать двоичные переменные в блоки по 8 или 16 штук, а переменные типа BYTE - парами. Редактор сохраняет новые переменные типов BOOL или BYTE, выделяя им байтовые участки памяти. Для всех остальных типов переменных выделяются участки памяти размером в 1 слово (начиная с байта с четным адресом).

Окно программы

В окне программы Вы увидите (в зависимости от установок редактора, принятых по умолчанию) поля для заголовка и комментария блока и, если это первый сегмент, то поля для заголовка и комментария сегмента, а также поле для ввода программы. В разделе программы кодового блока Вы можете управлять отображением комментариев и символов (имен) с помощью опций меню: *View -> Comment (Вид -> Комментарий)*, *View -> Symbolic Representation (Вид -> Представление символов)*, *View -> Symbol Information (Вид -> Информация о символе)*. Также Вы можете менять размер отображения с помощью опций меню: *View -> Zoom In (Вид -> Увеличить масштаб)*, *View -> Zoom Out (Вид -> Уменьшить масштаб) View -> Zoom Factor (Вид -> Коэффициент масштабирования)*.

Вы можете разделить STL-программу на сегменты. Редактор нумерует сегменты автоматически, начиная с 1. Вы можете дать каждому сегменту заголовок сегмента (*network title*) и комментарий сегмента (*network comment*). Во время редактирования Вы можете выбирать каждый сегмент непосредственно с помощью опций меню: *Edit -> Go To -> ...(Правка -> Перейти к ->…)*. Сегментирование не является обязательным приемом.

Для начала ввода программного кода щелкните один раз кнопкой мыши ниже поля для комментария сегмента или, если Вы установили режим отображения "Display with Comments" ("Отображать с комментариями"), тогда щелкните один раз кнопкой мыши ниже выделенного тенью поля для комментария. Перед Вами пустая рамка окна. Здесь Вы можете вводить программу в любом месте внутри этого окна. Обратитесь к разделу 3.4.1 "Структура STL-выражения" для получения информации о структуре STL-выражения.

Отделите оператор (OP-code [operator]) от адреса (операнд [operand]) одним или несколькими пробелами или одним шагом табуляции. После адреса в этой же строке Вы можете ввести две косых черты, после которых введите комментарий к выражению. Закончите выражение, нажав клавишу <Enter>. Вы можете также ввести целую строку комментария, начав новую строку с двойной косой черты "//".

Новый сегмент можно запрограммировать, выбрав опции меню: *Insert -> Network (Вставка -> Сегмент).* При этом редактор вставляет пустой сегмент после выбранного сегмента.

Если необходимо использовать символьные имена при инкрементном программировании, эти имена к моменту их использования в программе должны быть уже назначены абсолютным адресам.

Вы можете вызывать таблицу символов для выбора из нее символьных имен с помощью опций меню: *Insert -> Symbol (Вставка -> Символ)*. После вызова таблицы символов требуемый символ переносится в программу после щелчка на нем кнопкой мыши.

При инкрементном программировании Вы также можете вносить новые символьные имена в таблицу символов или корректировать имена, ранее в нее внесенные. Вы можете вызывать таблицу символов целиком с помощью опций меню: *Option -> Symbol Table (Опции -> Таблица символов)*, также Вы можете вызывать одну строку из таблицы символов для редактирования с помощью опций меню: *Edit -> Symbol (Правка -> Символ).* После редактирования или ввода нового символьного имени Вы можете использовать его, продолжив ввод своей программы.

Нет необходимости завершать блок специальным выражением. Тем не менее, Вы можете запрограммировать последний "пустой" сегмент с заголовком "Block End" ("Конец блока"), облегчая тем самым зрительное восприятие программы (это может быть полезно в случае особо длинных блоков).

Если с помощью редактора открывается ранее скомпилированный блок, этот блок "декомпилируется", т.е. для него генерируется STL-код. Для этого редактор использует разделы программы в базе данных программатора PG, которые не совсем соответствуют программе, например, в плане символов, комментариев и меток перехода. Если нужная информация в базе данных программатора PG отсутствует во время декомпиляции программы, редактор использует подставленные символы (substitute symbols).

Вы можете в редакторе создать новые блоки или открыть и отредактировать существующие без необходимости перехода в утилиту SIMATIC Manager.

Каталог элементов программы

Если каталог элементов программы не видим, активизируйте его с помощью опций меню: *View -> Catalog (Вид -> Каталог)*.

Каталог элементов программы располагается в своем собственном окне в правой части окна редактора. Каталог можно убрать с экрана, если дважды щелкнуть кнопкой манипулятора "мышь" на заголовке окна каталога.

Каталог элементов программы поддерживается при программировании на языках LAD и FBD, обеспечивая доступные графические элементы. При программировании на языке STL этот каталог показывает только блоки, которые находятся в автономном (offline) каталоге *Blocks (Блоки)*, а также уже запрограммированные экземпляры мультиэкземплярных FB и доступные библиотеки.

3.4.3 Программирование кодовых блоков на STL, ориентированное на создание исходных файлов

В разделе 2.5 "Создание S7-программ" Вы можете найти введение в основы создания S7-программы и использование программного редактора. Процесс программирования блока, ориентированный на создание исходных файлов, начинается с создания пустого файла исходной программы в SIMATIC Manager (см. раздел 2.5.3 "Редактор STL-программ [STL Program Editor] под подзаголовком "Программирование, ориентированное на создание исходных файлов").

Теперь Вы можете запустить редактор, открыв исходный файл, и можете немедленно начать вводить программу, например, с помощью ключевого слова для функционального блока.

В таблице 3.3 приведены ключевые слова, требующиеся при программировании блоков, а также порядок их использования.

Заголовок блока

Вы должны запрограммировать свойства блока в его заголовке после указания типа блока и перед разделом объявления переменных. Вся информация в заголовке блока опциональна (optional); Вы можете пропустить или отдельные характеристики, или все характеристики без исключения. Для получения информации по описанию и назначению свойств блоков обратитесь к разделу 3.2.3 "Свойства блоков".

С помощью ключевого слова "TITLE =" сразу же после строки с указанием типа блока Вы можете ввести заголовок блока длиной до 64 символов. Вы можете вслед за тем добавить комментарий в одной или нескольких строках, начинающихся двойной косой чертой. Комментарий блока может иметь размер до 18 Кбайт.

Описание переменных

Раздел объявления переменных содержит определения всех внутриблочных переменных, т.е. таких переменных, которые Вы применяете только в этом блоке. Вы не можете использовать любые типы переменных в любом блоке (см. таблицу 3.3). Если Вы не используете какие-либо типы переменных, пропустите соответствующие описания, включая ключевые слова.

Описание переменной состоит из имени, типа данных, значения по умолчанию (если есть) и комментария переменной (необязательный элемент).

Например:

Quantity : INT := +500; //Units per batch (единиц на пакет)

Таблица 3.3 Ключевые слова для программирования кодовых блоков на STL

Блок	Организационный блок	Функциональный блок	Функция
Тип блока	ORGANIZATION_BLOCK	FUNCTION_BLOCK	FUNCTION : *значение*
Заголовок [Header]	TITLE = *Заголовок блока* *//Комментарий блока* KNOW_HOW_PROTECT NAME : *Имя блока* FAMILY : *Второе имя* AUTHOR : *Автор* VERSION : *Версия*	TITLE = *Заголовок блока* *//Комментарий блока* CODE_VERSION1 KNOW_HOW_PROTECT NAME : *Имя блока* FAMILY : *Второе имя* AUTHOR : *Автор* VERSION : *Версия*	TITLE = *Заголовок блока* *//Комментарий блока* KNOW_HOW_PROTECT NAME : *Имя блока* FAMILY : *Второе имя* AUTHOR : *Автор* VERSION : *Версия*
Описание [Declaration]		VAR_INPUT *Входные параметры* END_VAR	VAR_INPUT *Входные параметры* END_VAR
		VAR_OUTPUT *Выходные параметры* END_VAR	VAR_OUTPUT *Выходные параметры* END_VAR
		VAR_IN_OUT *Вх/Вых параметры* END_VAR	VAR_IN_OUT *Вх/Вых параметры* END_VAR
		VAR *Статические локальные данные* END_VAR	
	VAR_TEMP *Временные локальные данные* END_VAR	VAR_TEMP *Временные локальные данные* END_VAR	VAR_TEMP *Временные локальные данные* END_VAR
Программа [Program]	BEGIN NETWORK TITLE = *Заголовок сегмента* *//Комментарий* *//сегмента* *…STL-операторы* *//Комментарий строки* NETWORK *и т.д.*	BEGIN NETWORK TITLE = *Заголовок сегмента* *//Комментарий* *//сегмента* *…STL-операторы* *//Комментарий строки* NETWORK *и т.д.*	BEGIN NETWORK TITLE = *Заголовок сегмента* *//Комментарий* *//сегмента* *…STL-операторы* *//Комментарий строки* NETWORK *и т.д.*
Конец блока [Block end]	END_ORGANIZATION_BLOCK	END_FUNCTION_BLOCK	END_FUNCTION

Не всем переменным могут быть назначены значения по умолчанию (не могут значения по умолчанию назначаться, например, для временных локальных данных). Назначения по умолчанию для функций и функциональных блоков детально описаны в главе 19 "Параметры блоков".

Порядок описаний в кодовых блоках является регламентированным (см. таблицу 3.3), в то же время порядок следования типов переменных в разделе описания переменных произволен, при этом порядок следования типов переменных влияет (с учетом типов данных) на расходование памяти. В главе 24 "Типы данных" показано, как оптимизировать распределение памяти, правильно планируя порядок размещения данных.

Раздел программы

Раздел программы кодового блока начинается с ключевого слова BEGIN и заканчивается ключевым словом END_xxx, в котором xxx заменяется, в зависимости от типа блока, на ORGANIZATION_BLOCK, FUNCTION_BLOCK или FUNCTION. Ключевое слово END_xxx заменяет Block End BE.

И в ключевых словах, и в тексте программы редактор различает, какой регистр используется (верхний или нижний). Подробнее о синтаксисе выражений Вы можете прочитать в разделе 3.4.1 "Структура STL-выражения". OP-код (оператор) может быть отделен от адреса (операнд) одним или несколькими пробелами или шагами табуляции. Для улучшения читаемости исходного текста программы Вы можете оставлять один или несколько пробелов (и/или шагов табуляции) между словами. Вы должны заканчивать каждое выражение точкой с запятой ";". После точки с запятой Вы можете записать комментарий, который должен начинаться с двойной косой черты "//". Комментарий может продолжаться до конца строки. Вы можете помещать несколько выражений в одной строке, разделяя их точкой с запятой ";". Вы также можете записывать комментарии с начала строки, помещая в начале строки двойную косую черту "//". Строка комментария не может содержать более 160 символов; она не может содержать символов табуляции и непечатаемых символов.

Для улучшения читаемости и логики программы Вы можете разбить программу блока на сегменты (*network*). В графических языках (с графической интерпретацией) такое разбиение обязательно, в языке STL - необязательно. Сегменты в STL не имеют функционального назначения; они используются здесь просто для разбиения программы на большее количество логически связанных частей и для улучшения ее читаемости, а также чтобы упростить и сделать более эффективным написание комментариев. В очень больших программах сегментирование программы дает преимущество, заключающееся в том, что возможен прямой доступ к сегментам в скомпилированном блоке, что способствует быстрому доступу к отдельным точкам в программе посредством опций: *Edit -> Go To -> ... (Правка -> Перейти к->…).* Так, Вы можете задавать номер сегмента или номер строки, относящийся к началу сегмента.

Сегменты начинаются с ключевого слова NETWORK; ключевое слово "TITLE=" в следующей строке позволяет задавать сегменту заголовок длиной до 64 символов. Строка комментариев, следующая сразу за заголовком, образует комментарий сегмента, вмещающий до 18 Кбайт информации. Редактор STL нумерует сегменты автоматически, начиная с номера 1. В каждом блоке может находиться до 999 сегментов. Всего пользователю доступно 64 Кбайта памяти для комментариев блока и сегмента в каждом блоке.

Порядок блоков при программировании, ориентированном на создание исходных файлов программы

Для вызова блока редактор требует информацию из заголовка блока, заданные параметры блока, заявленный тип, а также тип данных параметров блока. Это значит, что Вы должны сначала запрограммировать вызываемые функции и функциональные блоки, то есть Вы должны начать программирование программы с блоков "самого нижнего уровня" (имеется в виду положение блоков относительно начала программы в исходном файле).

Однако, достаточно запрограммировать заголовки блоков и задать их параметры (то есть задать описание интерфейса ["interface description"]), чтобы не было ошибки при вызове блока. В дальнейшем Вы можете снабдить этот "интерфейс" программой, Однако необходимо сохранять без изменения интерфейс уже вызванного блока! Иначе редактор выдаст сообщение о конфликте временных меток при вызове блока.

В случае большой пользовательской программы Вы будете, очевидно, разбивать исходный файл программы на отдельные легкоуправляемые файлы, например, на "стандартные подпрограммы", которые можно использовать неоднократно по всей программе, технологически- или функционально-законченные подпрограммы и главную программу (main program), которая содержит, как правило, организационные блоки. При создании отдельных исходных файлов, Вы должны следить за порядком компилирования, для соблюдения правил вызовов блоков, приведенных выше. Рекомендуется использовать следующий порядок компилирования:

- Пользовательские типы данных UDT
- Блоки глобальных данных
- Функции и функциональные блоки, начиная с блоков самого низкого уровня вызовов
- Экземплярные блоки данных (эти блоки могут также быть расположены непосредственно за назначенным функциональным блоком)
- Организационные блоки

Пример функционального блока с экземплярным блоком данных

На рисунке 3.8 показан пример функционального блока со статическими локальными данными, за которым следует экземплярный блок данных, связанный с этим функциональным блоком.

3.5 Программирование кодовых блоков на SCL

3.5.1 Структура SCL-выражения

SCL-программа состоит из ряда отдельных выражений (statement). Выражение - это наименьшая самостоятельная единица программы пользователя. Выражение содержит описание работы для CPU. На рисунке 3.9 показаны несколько примеров SCL-выражений.

В составе SCL-выражения можно выделить следующие компоненты:

- Метка перехода (необязательный элемент), содержащая до 24 символов и заканчивающаяся двоеточием ":". Метки перехода должны быть описаны.
- Инструкция, описывающая задание для CPU (например, присвоение значений, оператор управления и т.д.)
- Комментарий (необязательный элемент), начинающийся двойной косой чертой "//", может продолжаться до конца строки и содержать только печатаемые символы (кроме табуляции).

Каждое SCL-выражение должно завершаться точкой с запятой ";" (перед комментарием). SCL-выражение может содержать до 126 символов.

```
FUNCTION_BLOCK V_Memory
TITLE = Intermediate buffer  for 4 values
//В заголовке: промежуточный буфер на 4 значения
//Пример блока FB с параметрами и статическими локальными данными на STL
AUTHOR   : Berger
FAMILY   : STL_Book
NAME     : Memory
VERSION  : 01.00
VAR_INPUT
 Transfer    : BOOL := FALSE;//Пересылка положительного фронта сигнала
 Input_value : REAL := 0.0;  //в формате данных REAL  (дробное число)
END_VAR
VAR_OUTPUT
 Output_value : REAL := 0.0; //в формате данных REAL  (дробное число)
END_VAR
VAR
 Value1 : REAL := 0.0; //первое сохраненное значение в формате данных REAL
 Value2 : REAL := 0.0; //второе значение
 Value3 : REAL := 0.0; //третье значение
 Value4 : REAL := 0.0; //четвертое значение
 Edge_memory_bit : BOOL := FALSE;//меркер фронта передаваемого сигнала
END_VAR
BEGIN
NETWORK
TITLE = Program for transfer and output
//Передача и вывод имеют место, если в Transfer положительный фронт сигнала
  A     Transfer;        // если Transfer устанавливается в значение "1"
  FP    Edge_memory_bit; // RLO устанавливается в "1" вслед за FP
  JCN   End;             // переход, если нет положительного фронта
//Передача значений, начиная с последнего
  L     Value4;
  T     Output_value;    //Передача последнего значения
  L     Value3;
  T     Value4;
  L     Value2;
  T     Value3;
  L     Value1;
  T     Value2;
  L     Input_value;     //Передача входного значения
  T     Value1;
End: BE;
END FUNCTION BLOCK



DATA_BLOCK Values1
TITLE = Instance data block for "V_Memory"
//Пример экземплярного блока данных для FB "V_Memory"
AUTHOR    : Berger
FAMILY    : STL_Book
NAME      : V_MEM_DB1
VERSION   : 01.00
  V_Memory          //экземпляр для FB "V_Memory"
BEGIN
  Value1  := 1.0;   //Индивидуальные присвоения для
  Value2  := 1.3;   //отдельных значений
END_DATA_BLOCK
```

Рис. 3.8 Пример программирования функционального STL-блока со связанным экземпляром блока данных

```
Value Assignments      (присвоение значений)

Power      := Voltage * Current;
TooLarge   := Volt_Act > Volt_Set;
Switch_on  := Manual_on OR Auto_on;

Control Statements     (операторы управления)

IF Input_value > Maximum
  THEN Delimiter    := Maximum;
  ELSIF Input_value  < Minimum
    THEN Delimiter  := Minimum;
  ELSE Delimiter    := Input_value;
END_IF;
FOR i := 1 TO 32 DO
   Measure_value[i] := 0;
END_FOR;

Function Calls         (вызовы функций)

Result  := Delimiter(
   Input_value:= Actual_value,
   Minimum    := Lower_limit,
   Maximum    := Upper_limit)
```

Рис. 3.9 Примеры STL-выражений

3.5.2 Программирование кодовых SCL-блоков

В разделе 2.5 "Создание S7-программ" Вы можете найти введение в основы создания S7-программы и использование программного редактора.

Процесс программирования блока начинается с создания пустого файла исходной программы в SIMATIC Manager (см. раздел 2.5.4 "Редактор SCL-программ [SCL Program Editor] под подзаголовком "Программирование исходного SCL-файла").

Теперь Вы можете запустить редактор, открыв исходный файл, и можете немедленно начать вводить программу, например, с помощью ключевого слова для функционального блока.

В таблице 3.4 приведены ключевые слова, требующиеся при программировании блоков, а также порядок их использования.

Заголовок блока

Вы должны запрограммировать свойства блока в его заголовке после указания типа блока и перед разделом объявления переменных. Вся информация в заголовке блока опциональна (optional); Вы можете пропустить или отдельные характеристики, или все характеристики без исключения. Для получения информации по описанию и назначению свойств блоков обратитесь к разделу 3.2.3 "Свойства блоков".

С помощью ключевого слова "TITLE =" сразу же после строки с указанием типа блока Вы можете ввести заголовок блока длиной до 64 символов. Вы можете вслед за тем добавить комментарий в одной или нескольких строках, начинающихся двойной косой чертой "//". Комментарий блока может иметь размер до 18 Кбайт.

Таблица 3.4 Ключевые слова для программирования кодовых блоков на SCL

Блок	Организационный блок	Функциональный блок	Функция
Тип блока	ORGANIZATION_BLOCK	FUNCTION_BLOCK PROGRAM[3]	FUNCTION : *значение функции*
Заголовок [Header]	TITLE = *Заголовок блока* *//Комментарий блока* KNOW_HOW_PROTECT NAME : *Имя блока* FAMILY : *Второе имя* AUTHOR : *Автор* VERSION : *Версия*	TITLE = *Заголовок блока* *//Комментарий блока* CODE_VERSION1 KNOW_HOW_PROTECT NAME : *Имя блока* FAMILY : *Второе имя* AUTHOR : *Автор* VERSION : *Версия*	TITLE = *Заголовок блока* *//Комментарий блока* KNOW_HOW_PROTECT NAME : *Имя блока* FAMILY : *Второе имя* AUTHOR : *Автор* VERSION : *Версия*
Описание [Declaration]		VAR_INPUT *Входные параметры* END_VAR	VAR_INPUT *Входные параметры* END_VAR
		VAR_OUTPUT *Выходные параметры* END_VAR	VAR_OUTPUT *Выходные параметры* END_VAR
		VAR_IN_OUT *Вх/Вых параметры* END_VAR	VAR_IN_OUT *Вх/Вых параметры* END_VAR
		VAR *Статические локальные данные* END_VAR	VAR[1] *Временные локальные данные* END_VAR
	VAR_TEMP *Временные локальные данные* END_VAR	VAR_TEMP *Временные локальные данные* END_VAR	VAR_TEMP *Временные локальные данные* END_VAR
	CONST *Константы* END_CONST	CONST *Константы* END_CONST	CONST *Константы* END_CONST
	LABEL *Метки перехода* END_LABEL	LABEL *Метки перехода* END_LABEL	LABEL *Метки перехода* END_LABEL
Программа [Program]	BEGIN[2] *…SCL-операторы* *//Комментарий строки* (**Комментарий блока …* *…Комментарий блока*)* *… и т.д.*	BEGIN[2] *…SCL-операторы* *//Комментарий строки* (**Комментарий блока …* *…Комментарий блока*)* *… и т.д.*	BEGIN[2] *…SCL-операторы* *//Комментарий строки* (**Комментарий блока …* *…Комментарий блока*)* *… и т.д.*
Конец блока [Block end]	END_ORGANIZATION_ BLOCK	END_FUNCTION_BLOCK END_PROGRAM[3]	END_FUNCTION

[1] Локальные данные под ключевым словом VAR в SCL-функции FC используются как временные локальные данные (VAR_TEMP).

[2] Не требуется в SCL.

[3] Альтернативный вариант функциональному блоку: FUNCTION_BLOCK и END_FUNCTION_BLOCK.

Описание переменных

Раздел объявления переменных содержит определения всех внутриблочных переменных, т.е. таких переменных, которые Вы применяете только в этом блоке. Вы не можете использовать любые типы переменных в любом блоке (см. таблицу 3.4). Если Вы не используете какие-либо типы переменных, пропустите соответствующие описания, включая ключевые слова.

Описание переменной состоит из имени, типа данных, значения по умолчанию (если есть) и комментария переменной (необязательный элемент).

Например:

Quantity : INT := +500; //Units per batch (единиц на пакет)

При описании переменных допускается группировать переменные одного типа в одной строке, например:

Value1, Value2, Value3, Value4 : INT;

Не всем переменным могут быть назначены значения по умолчанию (не могут значения по умолчанию назначаться, например, для временных локальных данных). Назначения по умолчанию для функций и функциональных блоков детально описаны в главе 19 "Параметры блоков".

Порядок описаний в кодовых блоках является регламентированным (см. таблицу 3.4), в то же время порядок следования типов переменных в разделе описания переменных произволен, при этом порядок следования типов переменных влияет (с учетом типов данных) на расходование памяти. В главе 24 "Типы данных" показано, как оптимизировать распределение памяти, правильно планируя порядок размещения данных.

При программировании на SCL Вы можете объявлять константы, то есть можно назначать символическое имя фиксированным значениям.

Если в SCL-программе Вы используете метки перехода, то Вы также должны объявить их.

Раздел программы

Раздел программы кодового SCL-блока может начинаться с ключевого слова BEGIN (опционально) и заканчивается ключевым словом END_xxx, в котором xxx заменяется, в зависимости от типа блока, на ORGANIZATION_BLOCK, FUNCTION_BLOCK или FUNCTION. Ключевое слово END_xxx заменяет Block End (BE).

И в ключевых словах, и в тексте программы редактор различает, какой регистр используется (верхний или нижний). Подробнее о синтаксисе выражений Вы можете прочитать в разделе 3.5.1 "Структура SCL-выражения". OP-код (оператор) может быть отделен от адреса (операнд) одним или несколькими пробелами или шагами табуляции. Для улучшения читаемости исходного текста программы Вы можете оставлять один или несколько пробелов (и/или шагов табуляции) между словами.

Вы должны заканчивать каждое выражение точкой с запятой ";". После точки с запятой Вы можете записать комментарий, который должен начинаться с двойной косой черты "//". Комментарий может продолжаться до конца строки. Вы можете помещать несколько выражений в одной строке, разделяя их точкой с запятой ";".

SCL-блок должен содержать по крайней мере одно SCL-выражение (один знак точки с запятой ";"). SCL-программа не имеет сегментов в отличие от STL-программы.

Вы также можете записывать комментарии с начала строки, помещая в начале строки двойную косую черту "//". Строка комментария не может содержать более 160 символов; она не может содержать символов табуляции и непечатаемых символов.

В SCL Вы можете создавать комментарий блока, который может занимать несколько строк. Он начинается открывающей скобкой и звездочкой "(*" и заканчивается звездочкой и закрывающей скобкой "*)". Комментарий блока может также помещаться внутри SCL-выражения; однако он не может "разрывать" символических имен или констант (исключение: строка символов).

Порядок блоков при программировании, ориентированном на создание исходных файлов программы

Для вызова блока редактор требует информацию из заголовка блока, заданные параметры блока, заявленный тип, а также тип данных параметров блока. Это значит, что Вы должны сначала запрограммировать вызываемые функции и функциональные блоки, то есть Вы должны начать программирование программы с блоков "самого нижнего уровня" (имеется в виду положение блоков относительно начала программы в исходном файле).

Однако, достаточно запрограммировать заголовки блоков и задать их параметры (то есть задать описание интерфейса ["interface description"]), чтобы не было ошибки при вызове блока. В дальнейшем Вы можете снабдить этот "интерфейс" программой, Однако необходимо сохранять без изменения интерфейс уже вызванного блока! Иначе редактор выдаст сообщение о конфликте временных меток при вызове блока.

В случае большой пользовательской программы Вы будете, очевидно, разбивать исходный файл программы на отдельные легкоуправляемые файлы, например, на "стандартные подпрограммы", которые можно использовать неоднократно по всей программе, технологически- или функционально-законченные подпрограммы и главную программу (main program), которая содержит, как правило, организационные блоки. При создании отдельных исходных файлов, Вы должны следить за порядком компилирования, для соблюдения правил вызовов блоков, приведенных выше.

Рекомендуется использовать следующий порядок компилирования:

- Пользовательские типы данных UDT
- Блоки глобальных данных
- Функции и функциональные блоки, начиная с блоков самого низкого уровня вызовов
- Экземплярные блоки данных (эти блоки могут также быть расположены непосредственно за назначенным функциональным блоком)
- Организационные блоки

Пример функционального блока с экземплярным блоком данных

На рисунке 3.10 показан пример функционального блока со статическими локальными данными, за которым следует экземплярный блок данных, связанный с этим функциональным блоком.

```
FUNCTION_BLOCK V_Memory
TITLE = 'Intermediate buffer for 4 values'
//В заголовке: промежуточный буфер на 4 значения
//Пример блока FB с параметрами и статическими локальными данными на SCL
AUTHOR   : Berger
FAMILY   : SCL_Book
NAME     : Memory
VERSION  : 01.00

VAR_INPUT
 Transfer    : BOOL := FALSE;//Пересылка положительного фронта сигнала
 Input_value : REAL := 0.0;  //в формате данных REAL  (дробное число)
END_VAR

VAR_OUTPUT
 Output_value : REAL := 0.0; //в формате данных REAL  (дробное число)
END_VAR

VAR
 Value1 : REAL := 0.0; //первое сохраненное значение в формате данных REAL
 Value2 : REAL := 0.0; //второе значение
 Value3 : REAL := 0.0; //третье значение
 Value4 : REAL := 0.0; //четвертое значение
 Edge_memory_bit : BOOL := FALSE;//меркер фронта передаваемого сигнала
END_VAR

BEGIN
//Передача и вывод имеют место, если в Transfer положительный фронт

IF Transfer = 1 AND Edge_memory_bit = 0
THEN Output_value  := Value4;
     //Передача начинается с последнего значения
     Value4  := Value3;
     Value3  := Value2;
     Value2  := Value1;
     Value1  := Input_value;
     Edge_memory_bit  := Transfer;   //обновление меркера, даже
ELSE Edge_memory_bit  := Transfer;   //если нет фронта
END_IF;

END FUNCTION BLOCK


DATA_BLOCK Values1
TITLE = 'Instance data block for  "V_Memory"
//Пример экземплярного блока данных для FB "V_Memory"

AUTHOR    : Berger
FAMILY    : SCL_Book
NAME      : V_MEM_DB1
VERSION   : 01.00
  V_Memory          //экземпляр для FB "V_Memory"
BEGIN
  Value1  := 1.0;   //Индивидуальные присвоения для
  Value2  := 1.3;   //отдельных значений
END_DATA_BLOCK
```

Рис. 3.10 Пример программирования функционального SCL-блока со связанным экземплярным блоком данных

3.6 Программирование блоков данных

В разделе 2.5 "Создание S7-программ" Вы можете найти введение в основы создания S7-программы и использование программного редактора.

Блоки данных программируются на языках программирования STL и SCL таким же образом, что и кодовые блоки. Для инкрементного программирования Вы будете использовать редактор STL-программ. Для программирования, ориентированного на создание исходных текстовых файлов программ используются как редактор STL-программ, так и редактор SCL-программ.

3.6.1 Инкрементное программирование блоков данных

Создание блоков

Процесс программирования блока начинается с его открытия одним из двух способов: либо двойным щелчком на блоке в окне проекта SIMATIC Manager, либо в редакторе STL-программ с помощью выбора опций меню: *File -> Open (Файл -> Открыть).* Если блок пока не существует, Вы должны его сначала создать одним из следующих путей:

- В левой половине окна проекта SIMATIC Manager выберите объект *Blocks (Блоки),* создайте новый блок данных с помощью опций меню: *Insert -> S7 Block -> Data Block (Вставка -> S7 Block -> Блок данных)*. В окне свойств (Properties) блока на вкладке "General - Part 1" ("Общие - часть 1") выберите номер блока. Язык программирования (creation language) установлен как "DB". Вы можете ввести остальные свойства блока позже.
- Находясь в редакторе, с помощью опций меню: *File -> New (Файл -> Создать)* вызовите окно диалога, в котором Вы можете задать требуемый блок в окне "Object name" ("Имя объекта"). После закрытия диалогового окна Вы можете вводить программу этого блока.

Вы можете ввести информацию заголовка блока либо при создании блока, либо позднее, активизировав редактор, затем открыв блок и выбрав опции меню: *File -> Properties (Файл -> Свойства)*.

Типы блоков данных

Когда Вы впервые открываете новый блок данных, Вы увидите окно "New Data Block" ("Новый блок данных"); теперь Вы должны решить, какой тип назначать для создаваемого блока.

Вы можете назначить блоку данных один из трех следующих типов, щелкнув кнопкой манипулятора "мышь" по одному из них:

- "Data Block" ("Блок данных")
  При выборе данной опции создается блок глобальных данных; в данном случае Вы должны объявить назначенные адреса данных при программировании блока.
- "Data block with assigned user-defined data type" ("Блок данных пользовательского типа")
  При выборе данной опции создается блок данных пользовательского типа; в данном случае Вы должны объявить структуру данных пользовательского типа UDT.

- "Data block with assigned function block" ("Блок данных с назначением функциональному блоку")
  При выборе данной опции создается экземплярный блок данных; в данном случае Вы должны объявить структуру данных для пересылки в соответствующий функциональный блок.

Окно блока

На нижеприведенном рисунке представлен пример открытого блока данных (см. рис. 3.11).

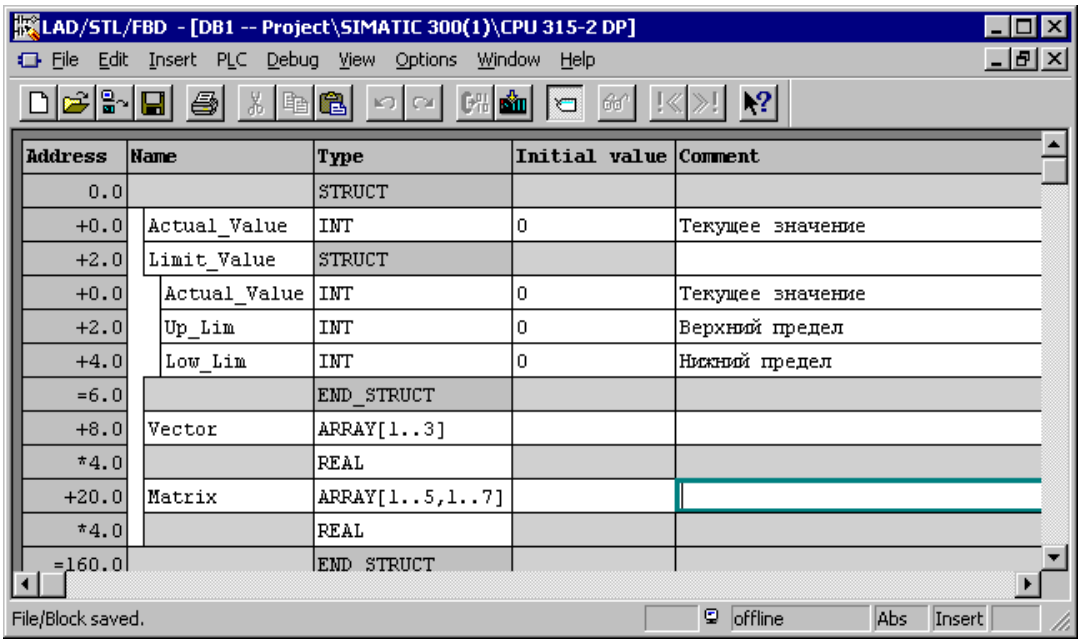


Рис.3.11 Пример открытого блока данных (Declaration View [вид объявления данных])

Вы можете выбирать один из двух видов окна:

- Declaration View (вид объявления данных)
  При выборе данной опции Вы вводите назначенные адреса данных, задаете тип данных и определяете начальные значения.
- Data View (вид фактических значений данных)
  При выборе данной опции редактор отображает фактические значения данных, которые Вы можете редактировать.

При программировании блока глобальных данных Вы можете задать для каждого адреса начальное значение. Переменные имеют стандартное значение, принимаемое по умолчанию, равное нулю (0), наименьшему значению или пробелу (blank), в зависимости от типа данных.

Экземплярный блок данных, генерирующийся для функционального блока, принимает в качестве начальных значений данных значения, принятые по умолчанию, из раздела объявления переменных этого FB.

Блок, создаваемый из данных пользовательского типа UDT, принимает в качестве начальных значений данных значения, принятые по умолчанию, в UDT.

Редактор может отображать блок данных в двух видах:

Declaration View (вид объявления данных: *View -> Declaration View [Вид -> Вид объявления данных]*)
При использовании данного вида Вы можете определять адреса данных, кроме того Вы можете наблюдать переменные в том виде, который Вы определили для них, например, поле или пользовательский тип данных как одну переменную.

Data View (вид фактических значений данных: *View -> Data View [Вид -> Вид фактических значений данных]*)
При использовании данного вида редактор отображает каждую переменную и каждый компонент поля или структуры индивидуально. Данный вид обеспечивает дополнительно отображение столбца "Actual value" ("Фактическое значение"). Фактическое значение адреса данных - это то значение, которое это адрес имеет или будет иметь в основной (main) памяти CPU. Редактор использует здесь начальное значение в качестве значения по умолчанию.

Вы можете модифицировать фактические значения отдельно для каждого адреса данных.

Пример: допустим Вы создаете несколько экземплярных блоков данных для функционального блока. Тем не менее, Вам необходимо для каждого вызова функционального блока (для каждой пары FB/DB) иметь слегка отличающийся в предустановках отдельный экземплярный блок данных. Вы можете редактировать каждый блок данных, выбрав опции: *View -> Data View (Вид -> Вид фактических значений данных)* и затем задавая значения, действительные только для данного блока данных, в столбце "Actual value" ("Фактическое значение"). С помощью опций меню: *Edit -> Initialize Data Block (Правка -> Инициализировать блок данных)* Вы можете вновь вернуть начальные значения данным этого блока.

3.6.2 Программирование блоков данных, ориентированное на создание исходных файлов

При создании исходных файлов блоков данных Вы должны придерживаться структуры или порядка, показанного в таблице 3.5. Данная таблица регламентирует использование ключевых слов при программировании блоков данных. Это касается как исходных файлов STL-программ, так и исходных файлов SCL-программ.

Заголовок блока

Вы должны запрограммировать свойства блока в его заголовке после указания типа блока и перед разделом объявления переменных. Вся информация в заголовке блока опциональна (optional); Вы можете пропустить или отдельные характеристики, или все характеристики без исключения. Для получения информации по описанию и назначению свойств блоков обратитесь к разделу 3.2.3 "Свойства блоков".

С помощью ключевого слова "TITLE =" сразу же после строки с указанием типа блока Вы можете ввести заголовок блока длиной до 64 символов. Вы можете вслед за тем добавить комментарий в одной или нескольких строках, начинающихся двойной косой чертой. Комментарий блока может иметь размер до 18 Кбайт.

Таблица 3.5 Ключевые слова для программирования блоков данных

Блок	Блок глобальных данных [Global Data Block]	Блок глобальных данных из UDT [Global Data Block from UDT]	Экземплярный блок данных [Instance Data Block]
Тип блока	DATA_BLOCK	DATA_BLOCK	DATA_BLOCK
Заголовок [Header]	TITLE = *Заголовок блока* *//Комментарий блока* KNOW_HOW_PROTECT NAME : *Имя блока* FAMILY : *Второе имя* AUTHOR : *Автор* VERSION : *Версия* READ_ONLY UNLINKED	TITLE = *Заголовок блока* *//Комментарий блока* KNOW_HOW_PROTECT NAME : *Имя блока* FAMILY : *Второе имя* AUTHOR : *Автор* VERSION : *Версия* READ_ONLY UNLINKED	TITLE = *Заголовок блока* *//Комментарий блока* KNOW_HOW_PROTECT NAME : *Имя блока* FAMILY : *Второе имя* AUTHOR : *Автор* VERSION : *Версия*
Описание [Declaration]	STRUCT *Name : Type := Default* *(Имя : Тип := Значение по умолчанию)* END_STRUCT	*UDTname (имя UDT)*	*FBname (имя FB)*
Инициа-лизация [Initialization]	BEGIN *Name := Default* *(Имя := Значение по умолчанию)* *… и т.д.*	BEGIN KOMPname := Default (имя := Значение по умолчанию) *… и т.д.*	BEGIN KOMPname := Default (имя := Значение по умолчанию) *… и т.д.*
Конец блока [Block end]	END_DATA_BLOCK	END_DATA_BLOCK	END_DATA_BLOCK

Описание переменных

Раздел объявления переменных содержит определения всех внутриблочных переменных, т.е. таких переменных, которые Вы применяете только в этом блоке. Вы можете объявить блок данных как блок глобальных данных с отдельными ("individual") переменными, как блок глобальных данных с UDT и как экземплярный блок данных.

Описание переменной в блоке глобальных данных состоит из имени, типа данных, значения по умолчанию (если есть) и комментария переменной (необязательный элемент).

Например:

Quantity : INT := +500; //Units per batch (единиц на пакет)

Всем переменным могут быть назначены значения по умолчанию. Порядок следования переменных в разделе описания переменных произволен, при этом порядок следования типов переменных влияет (с учетом типов данных) на расходование памяти. В главе 24 "Типы данных" представлены требования к памяти для разных типов переменных. Из раздела 26.2 "Хранение данных в переменных" Вы можете получить информацию о том, как переменные хранятся в блоках данных. Вы можете оптимизировать распределение памяти, правильно планируя порядок размещения данных.

Если Вы не назначите других значений по умолчанию для переменных, редактор запишет в них нулевые значения (0), минимальные значения или заполнит их пробелами (blank), в зависимости от типа данных.

Блок, создаваемый из данных пользовательского типа UDT, состоит только из UDT. Вы можете использовать абсолютные адреса (например, UDT 51) или символьные адреса (например, "Frame header").

Экземплярный блок данных, генерирующийся для функционального блока, принимает в качестве начальных значений данных только значения из раздела объявления переменных этого FB или с абсолютной, или с символьной адресацией.

Инициализация блока данных

Раздел инициализации блока данных начинается с ключевого слова BEGIN и заканчивается ключевым словом END_DATA_BLOCK. Даже если Вы не назначаете значения по умолчанию для переменных в разделе инициализации, Вы должны ввести эти ключевые слова.

Значения, которые Вы определили в разделе инициализации блока данных соответствуют фактическим (actual) значениям при инкрементном программировании. При компилировании блока данных значения, принятые по умолчанию (default) для переменных в разделе инициализации блока, становятся начальными (initial) значениями этих переменных, а начальные значения становятся фактическими (actual) значениями. Если блок данных загружается в CPU, начальные (initial) значения пересылаются в загрузочную (load) память, а фактические (actual) значения пересылаются в рабочую (work) память CPU (см. раздел 2.6.5 "Обработка блоков" пункт "Блоки данных в автономном (offline)/в интерактивном (online) режимах").

Если Вы не определили начальные значения для переменных, редактор будет использовать начальные значения как фактические. Если Вы используете пользовательские типы данных со значениями, принимаемыми по умолчанию, в разделе объявления переменных, Вы можете переписать (заменить - "overwrite") эти значения, принимаемые по умолчанию, в разделе инициализации.

Это же касается экземплярных блоков данных, которые назначены функциональным блокам (со своими значениями, принимаемыми по умолчанию) как структуры данных. Здесь Вы можете сформировать фактические (actual) значения по отдельности для каждого экземпляра (для вызова функционального блока с конкретным блоком данных).

3.7 Переменные и константы

3.7.1 Общие замечания по поводу переменных

Переменная - это величина особого формата (см. рис. 3.12). Простые переменные состоят из адреса (например, адрес входа input 5.2) и типа данных (например, BOOL для дискретного [двоичного] значения). Адрес, в свою очередь, содержит идентификатор адреса (такой как I для входа input) и указание абсолютного месторасположения (такое, например, как 5.2, что означает 2-ой бит 5-ого байта). Вы можете также использовать для доступа к адресу или переменной символьное имя, назначив для этого адреса имя (символ) в таблице символов (Symbol Table).

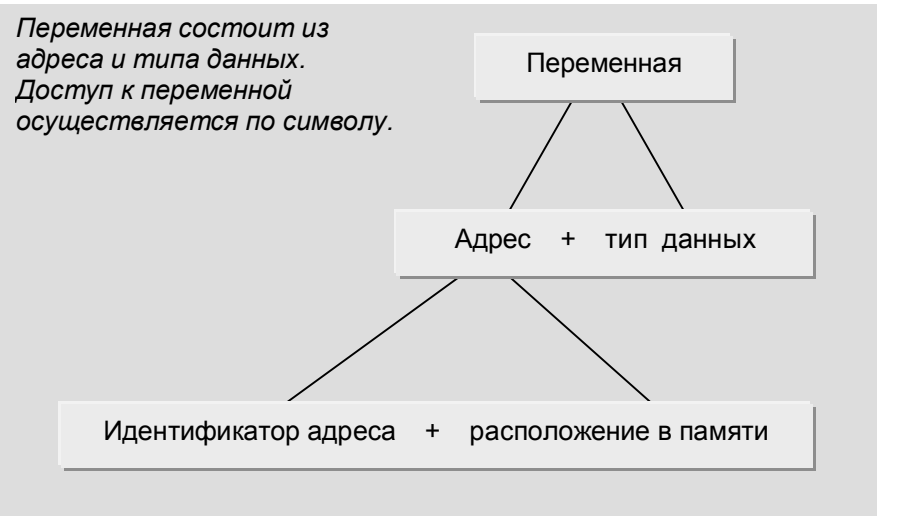


Рис. 3.12 Структура переменной

Бит данных типа BOOL адресуется как *двоичный адрес* (*binary address*) или *двоичный операнд* (*binary operand*). Адреса, содержащие один, два или четыре бита или переменные соответствующих типов называются *численными операндами (digital operand)*.

Переменные, объявленные внутри блока, являются внутриблочными (local - локальными) переменными. К ним относятся параметры блока, статические и временные локальные данные и даже адреса данных в блоках глобальных данных. Если локальные переменные являются переменными простого типа, они также могут быть доступны как операнды (например, статические локальные данные - как DI операнды, временные локальные данные - как L операнды, и данные в блоках глобальных данных - как DB операнды).

Тем не менее, локальные переменные могут быть сложных типов (например, такие переменные как структура или массив). Переменные этих типов требуют для размещения памяти больше, чем 32 бита, так что они не могут быть, например, загружены в аккумулятор. И по этой же причине они не могут быть адресованы с помощью обычного ("normal") STL-выражения. Существуют специальные функции для обработки таких переменных, такие как IEC-функции, которые поставляются в виде стандартной библиотеки с ПО STEP 7 (Вы можете создавать переменные сложных типов в параметрах блоков такого же типа).

Если переменные сложного типа содержат компоненты простого типа, то эти компоненты могут расматриваться, как если бы они были отдельными переменными (например, Вы можете загрузить в аккумулятор элемент массива, состоящего из 30 значений целого (INT) типа, и в дальнейшем обрабатывать его).

Константы используются для задания переменным фиксированных значений. Константа имеет особый префикс в зависимости от типа данных, к которому она принадлежит.

3.7.2 Общие замечания по поводу типов данных

Тип данных обуславливает характеристики данных, особенно это касается представления содержания переменной и диапазона допустимых для нее значений. STEP 7 предусматривает предопределение типов данных. Вы можете комбинировать типы данных, формируя пользовательске типы данных (User Data Type - UDT). Типы данных доступны в базовом пакете (Global Basis) и могут использоваться в каждом блоке.

В данном разделе приводится обзор всех типов данных. В разделе также содержится краткое введение в простые типы данных. Эти знания помогут Вам программировать PLC.

Ниже в таблице 3.6 приведен обзор типов данных в STEP 7.

Таблица 3.6 Классификация типов данных

Elementary Data Types (Простые типы данных)	Complex Data Types (Сложные типы данных)	User Data Types (Пользовательские типы данных)	Parametr Data Types (Типы данных для параметров)
BOOL, BYTE, CHAR, WORD, INT, DATE, DWORD, DINT, REAL, S5TIME, TIME, TOD	DT, STRING, ARRAY, STRUCT	UDT, блоки глобальных данных, экземпляры (Instances)	TIMER, COUNTER, BLOCK_DB, BLOCK_SDB, BLOCK_FC, BLOCK_FB, POINTER, ANY
Типы данных с объемом не более одного двойного слова (32 бита)	Типы, которые могут содержать данные объемом более одного двойного слова (DT, STRING) или состоящие из нескольких компонентов	Структуры или области данных, которые могут быть адресованы по имени	Параметры блока
Могут распределяться для операндов абсолютной или символьной адресацией	Могут распределяться только для переменных с символьной адресацией		Могут распределяться только для параметров блоков (только с символьной адресацией)
Разрешенные во всех адресных областях	Разрешенные в блоках данных (как глобальные данные и экземплярные данные), как временные локальные данные и как параметры блоков		Разрешенные в связи с параметрами блоков

3.7.3 Простые типы данных

Переменные простых типов данных могут быть отредактированы непосредственно в редакторе STL-программ, так как их значения могут быть занимать от одного до 32 битов в памяти. При присвоении значений в программе переменные простых типов данных также могут быть отредактированы непосредственно в SCL-редакторе.

Переменные простых типов могут быть предопределены фиксированными значениями (константами) на этапе объявления переменных.

Здесь записи в STL-программе (см. табл. 3.7) и записи в SCL-программе (см. табл. 3.8) отличаются друг от друга. Для многих типов данных существует больше одной формы для записи константы и все эти записи одинаково правомочны в употреблении (например, можно использовать форму записи TIME# или форму записи T#).

Запись константы в STL

Язык STL не накладывает ограничений на обработку (на операторы) типов данных (за исключением различий для двоичных [binary] и численных [digital] операндов). Функции сравнения, такие например, как сравнение содержимого аккумуляторов, не зависят от типа переменных, содержащихся в аккумуляторах.

Запись константы в SCL

При использовании языка SCL Вы можете обрабатывать переменные только разрешенных типов данных. Константы в SCL не принимают своего типа данных, пока они не будут поставлены в соответствие оператору.

Пример: в SCL константа 12345 относится к классу типов ANY_NUM, так как в зависимости от применения она будет иметь тип INT или DINT, или REAL. С помощью "определяющих тип" ("type-defined") записей Вы можете назначить отдельный тип данных непосредственно константе, например, с помощью определения DINT#12345 вы задаете для константы тип DINT.

3.7.4 Сложные типы данных

Вы можете использовать сложные типы данных (табл. 3.9) для работы с переменными в блоках данных или в L-стеке, а также для работы с параметрами блока.

Переменные сложных ("complex") типов, которые назначаются параметрам блоков, могут быть только полными ("complete") переменными; отдельные части переменных сложных типов не могут быть обработаны с помощью обычных ("normal") операторов. Тем не менее с помощью прямого доступа к переменным ("direct variable access") и с помощью косвенной адресации STL обеспечивает способ управления переменными, если известна их внутренняя структура.

Кроме того существуют IEC-функции, с помощью которых можно обрабатывать переменные DT и STRING (например, объединение двух символьных строк в одну). IEC-функции являются составной частью системы STEP 7; Вы можете найти их в стандартной библиотеке "Standard Library" в разделе "IEC Function Blocks". IEC-функции могут быть использованы в любом языке программирования. Длина переменной DT является фиксированной; Вы можете сами задавать длину переменных STRING, ARRAY и STRUCT при их определении.

Строка символов STRING может содержать до 254 символов и резервирует в памяти на 2 байта больше, чем число символов в строке.

Массив может иметь до 65536 элементов для каждого из индексов (т.е. от -32768 до 32767).

Таблица 3.7 Обзор простых типов данных при записи констант на языке STL

Data Type (width) (Тип данных, размер)	Description (Описание)	Пример записи STL-констант	
		минимальное значение	максимальное значение
BOOL (1 бит)	Bit (двоичное число)	FALSE (ЛОЖЬ)	TRUE (ИСТИНА)
BYTE (8 битов)	8-bit hexadecimal number (8-разрядное шестнадцатеричное число)	B#16#00, 16#00	B#16#FF, 16#FF
CHAR (8 битов)	One character (ASCII) (Один символ ASCII)	Печатный символ, например, 'A'	Печатный символ, например, 'A'
WORD (16 битов)	16-bit hexadecimal number (16-разрядное шестнадцатеричное число)	W#16#0000, 16#0000	W#16#FFFF, 16#FFFF
	16-bit binary number (16-разрядное двоичное число)	2#0000_0000_0000_0000	2#1111_1111_1111_1111
	Count value, 3 decades BCD (Значение счетчика, 3 декады формата BCD)	C#000	C#999
	Two 8-bit unsigned decimal numbers (Два 8-разрядных десятичных числа без знака)	B(0,0)	B(255,255)
DWORD (32 бита)	32-bit hexadecimal number (32-разрядное шестнадцатеричное число)	DW#16#0000_0000, 16#0000_0000	DW#16#FFFF_FFFF, 16#FFFF_FFFF
	32-bit binary number (32-разрядное двоичное число)	2#0000_0000...0000_0000	2#1111_1111...1111_1111
	Four 8-bit unsigned decimal numbers (Четыре 8-разрядных десятичных числа без знака)	B(0,0,0,0)	B(255,255,255,255)
INT (16 битов)	Fixed-point number (Число с фиксированной запятой)	-32 768	+32 767
DINT (32 бита)	Fixed-point number (Число с фиксированной запятой)	L# -2 147 483 648 [1]	L#+2 147 483 647 [1]
REAL (32 бита)	Floating-point number (Число с плавающей запятой)	в экспоненциальном представлении: +1.234567E+02 [2]	
		как десятичное число: 123.4567 [2]	
S5TIME (16 битов)	Time value in SIMATIC format (Значение времени в SIMATIC - формате)	S5T#0ms, S5TIME#0ms	S5T#2h46m30s, S5TIME#2h46m30s
TIME (32 бита)	Time value in IEC format (Значение времени в IEC-формате)	T#-24d20h31m23s647ms, TIME#-24d20h31m23s647ms	T#24d20h31m23s647ms, TIME#24d20h31m23s647ms
		T#-24.855134d, TIME#-24.855134d	T#24.855134d, TIME#24.855134d
DATE (16 битов)	Date (Дата)	D#1990-01-01, DATE# 1990-01-01	D#2168-12-31, DATE#2168-12-31
TIME_OF_DAY (32 бита)	Time of day (Суточное время)	TOD#00:00:00, TIME_OF_DAY#00:00:00	TOD#23:59:59.999, TIME_OF_DAY#23:59:59.999

[1] "L#" может быть опущено, если число за пределами диапазона целых чисел INT

[2] информация о диапазоне значений в разделе 24.1.3, "Представление чисел"

Таблица 3.8 Обзор простых типов данных при записи констант на языке SCL

Data Type (width) (Тип данных, размер)	Description (Описание)	Пример записи SCL-констант
BOOL (1 бит)	Bit (бит, двоичное число)	FALSE (ЛОЖЬ), TRUE (ИСТИНА), BOOL#FALSE, BOOL'TRUE. 2#0. 2#1. BOOL#0. BOOL#1
BYTE (8 битов)	8-bit decimal number 8-bit hexadecimal number 8-bit octal number 8-bit binary number Соответственно, 8-разрядные: десятичное, шестнадцатеричное, восьмеричное и двоичное числа	0, B#127, BYTE#255 16#0, B#16#7F, BYTE#16#FF 8#0, B#8#177, BYTE#8#377 2#0, B#2#0111_1111, BYTE#2#0111_1111
CHAR (8 битов)	One printable character (ASCII) (Печатный символ ASCII)	' ', CHAR#' ', CHAR#20 'z', CHAR#'z', CHAR#122
WORD (16 битов)	16-bit decimal number 16-bit hexadecimal number 16-bit octal number 16-bit binary number Соответственно, 16-разрядные: десятичное, шестнадцатеричное, восьмеричное и двоичное числа	0, W#32767, WORD#65535 16#0, W# 16#7FFF, WORD#16#FFFF 8#0, W#8#7J7777, WORD#8#17_7777 2#0, W#2#0111_1111_ …, WORD#2#1111_1111_…
DWORD (32 бита)	32-bit decimal number 32-bit hexadecimal number 32-bit octal number 32-bit binary number Соответственно, 32-разрядные: десятичное, шестнадцатеричное, восьмеричное и двоичное числа	0, DW#2147483647, DWORD#4294967295 16#0, DW#16#7FFF_FFFF, DWORD#16#FFFF_FFFF 8#0, DW#8#177_7777_7777, DWORD#8#377_7777_ ... 2#0, DW#2#0111_1111_... , DWORD#2#1111_1111_...
INT (16 битов)	16-bit decimal number 16-bit hexadecimal number 16-bit octal number 16-bit binary number Соответственно, 16-разрядные: десятичное, шестнадцатеричное, восьмеричное и двоичное числа	-32_768, 0, +32_767 INT#16#0, INT#16#7FFF, INT#16#FFFF INT#8#0, INT#8#7_7777, INT#8#17_7777 INT#2#0, INT#2#0111_1111_ ..., INT#2#1111_1111_...
DINT (32 бита)	32-bit decimal number 32-bit hexadecimal number 32-bit octal number 32-bit binary number Соответственно, 32-разрядные: десятичное, шестнадцатеричное, восьмеричное и двоичное числа	-2_147_483_648, 0, +2_147_483_647 DINT#16#0, DINT#16#7FFF_FFFF, DINT#16#FFFF_... DINT#8#0, DINT#8#177_7777_7777, DINT#8#377_... DINT#2#0, DINT#2#0111_1111_... , DINT#2#1111_...
REAL (32 бита)	Floating-point number (Число с плавающей запятой)	в экспоненциальном представлении: +1.234567E+02 [1)] как десятичное число: -123.4567 [1)] как целое число: +1234567 [1)]
S5TIME (16 битов)	Time value for SIMATIC timer functions (Значение времени в SIMATIC -формате)	T#0ms, TIME#2h46m30s T#0.0s, TIME#24.855134d
TIME (32 бита)	Time value in IEC format (Значение времени в IEC-формате)	T#-24d20h31m23s647ms, T#0ms, TIME#24d20h31m23s647ms T#-24.855134d, T#0.0ms, TIME#24.855134d
DATE (16 битов)	Date (Дата)	D# 1990-01-01, D#2168-12-31 DATE# 1990-01-01, DATE#2168-12-31
TIME_OF_DAY (32 бита)	Time of day (Суточное время)	TOD#00:00:00, TOD#23:59:59:999 TIME_OF_DAY#00:00:00, TIME_OF_DAY#23:59:59:999

[1)] информация о диапазоне значений в разделе 24.1.3, "Представление чисел"

Таблица 3.9 Обзор сложных типов данных

Тип данных	Описание		Пример
DATE_AND_TIME	Дата и время	64 бита	DT#1990-01-01-00:00:00.000 DATE_AND_TIME#2168-12-31:23:59:59.999
STRING	Строка символов	Переменная	Набор ASCII символов, например, "String 1"
ARRAY	Массив	Переменная	Набор компонентов одного типа данных; возможно до 6 размерностей
STRUCT	Структура	Переменная	Набор компонентов разного типа данных; возможно до 6 уровней вложения

3.7.5 Параметрические типы

Параметрические типы - это типы данных для параметров блоков (см. табл. 3.10). Размеры данных в таблице соответствуют области памяти, требуемой для параметров блока (для случая функциональных блоков). Используйте также TIMER и COUNTER в таблице символов как типы данных для таймеров и счетчиков.

Таблица 3.10 Обзор параметрических типов

Тип параметра	Описание		Пример фактических адресов
TIMER	Таймер	16 битов	T 15 или символ
COUNTER	Счетчик	16 битов	C 16 или символ
BLOCK_FC	Функция	16 битов	FC 17 или символ
BLOCK_FB	Функциональный блок	16 битов	FB 18 или символ
BLOCK_DB	Блок данных	16 битов	DB 19 или символ
BLOCK_SDB	Системный блок данных	16 битов	(до сих пор не используется)
POINTER	DB указатель	48 битов	Как указатель: P#M10.0 или P#DB20.DBX22.2 Как адрес: MW 20 или I 1.0 или символ
ANY	ANY указатель	80 битов	Как область: P#DB10.DBX0.0 WORD 20 или любая целая переменная