Основы программирования, соглашения и функции

5

S7–200 последовательно обрабатывает вашу программу для управления задачей или процессом. Программа создается с помощью STEP 7-Micro/WIN, а затем загружается в S7–200. STEP 7-Micro/WIN предоставляет в ваше распоряжение различные инструментальные средства и функции для разработки, реализации и тестирования вашей программы.

В этой главе

Рекомендации по проектированию системы автоматизации с микроконтроллером

Есть много методов проектирования систем с микроконтроллером. Следующие общие рекомендации применимы ко многим проектам. Конечно, вы должны следовать предписанным процедурам вашей собственной компании и учитывать ваш собственный опыт.

Расчленение процесса или машины

Разделите ваш процесс или машину на участки, не зависящие друг от друга. Эти участки определяют границы между несколькими системами автоматизации и влияют на описания функциональных областей и назначение ресурсов.

Описание функциональных областей

Запишите описания работы каждого участка процесса или машины. Включите следующие пункты: входы/выходы, описание функционирования, состояния, которые должны достигаться, перед тем как станет возможным управление исполнительными механизмами (например, выключателями с соленоидным приводом, двигателями и приводами), описание интерфейса оператора и интерфейсов с другими участками процесса или машинами.

Проектирование схем защиты

Определите оборудование, требующее для обеспечения безопасности аппаратно реализованной логики. Устройства управления могут выходить из строя опасным образом, вызывая неожиданный запуск или изменение в работе машинного оборудования. Там, где неожиданная или неправильная работа машинного оборудования может привести к физической травме людей или значительному материальному ущербу, нужно уделить внимание использованию электромеханических блокировок, которые работают независимо от S7–200, чтобы предотвратить опасные операции. В проектирование схем защиты должны включаться следующие задачи:

- Выявление ненадлежащей или неожиданной работы исполнительных механизмов, которая может оказаться опасной.
- Определение состояний, которые гарантировали бы, что работа не опасна, и выяснение того, как обнаруживать эти состояния независимо от S7–200.
- Определение влияния CPU S7–200 и входов/выходов на процесс при подаче и выключении питания и обнаружении ошибок. Эта информация должна использоваться только для проектирования нормального и ожидаемого аварийного режимов работы и не должна использоваться для целей безопасности.
- Проектирование ручных или электромеханических блокировок, которые блокируют опасную операцию независимо от S7–200.
- Предоставление в S7–200 надлежащей информации о состоянии от независимых цепей тока, чтобы программа и любые интерфейсы оператора имели необходимую информацию.
- Определение любых других связанных с безопасностью требований для безопасного протекания процесса.

Определение станций оператора

Основываясь на требованиях из описаний функциональных областей, разработайте чертежи станций оператора. Включите следующие пункты:

- Обзор, показывающий местоположение каждой станции оператора относительно процесса или машины
- Механическая компоновка устройств станции оператора, например, дисплеев, переключателей и ламп
- Электрические чертежи CPU S7–200 или модулей расширения с соответствующими входами-выходами

Разработка чертежей конфигурации

Основываясь на требованиях из описаний функциональных областей, разработайте чертежи конфигурации аппаратуры управления. Включите следующие пункты:

- Обзор, показывающий местоположение каждого S7–200 относительно процесса или машины
- Механическая компоновка S7–200 и модулей расширения входов-выходов (включая шкафы и другое оборудование)
- Электрические чертежи для каждого S7–200 и модуля расширения входов-выходов (включая номера моделей устройств, коммуникационные адреса и адреса входов-выходов)

Создание списка символических имен (не обязателен)

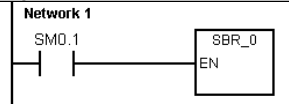
Если вы выбираете для адресации использование символических имен, то составьте список символических имен для абсолютных адресов. Включите не только сигналы физических входов-выходов, но также и другие элементы, которые нужно использовать в вашей программе.

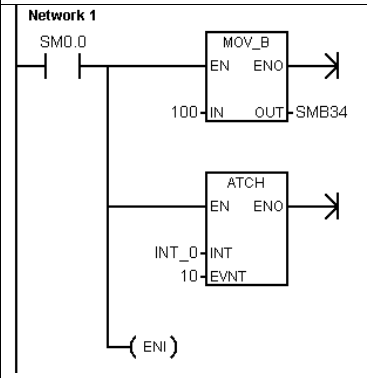
Основные элементы программы

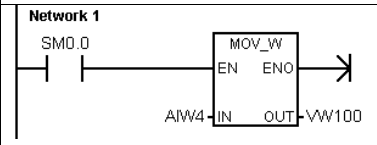
Программный блок состоит из исполняемого кода и комментариев. Исполняемый код состоит из основной программы (OB1), а также подпрограмм и программ обработки прерываний. Код компилируется и загружается в S7–200. Комментарии не компилируются и не загружаются. С помощью этих организационных элементов (основной программы, подпрограмм и программ обработки прерываний) вы можете структурировать свою управляющую программу.

В следующем примере показана программа, включающая в себя подпрограмму и программу обработки прерываний. Эта программа-пример с помощью прерывания, управляемого временем, считывает значение с аналогового входа каждые 100 мс.

Пример: Основные элементы программы

MAIN		Network 1 //В 1-ом цикле вызвать подпрограмму 0. LD SM0.1 CALL SBR_0

SBR0		Network 1 //Установить для прерывания,
		//управляемого временем,
		// интервал 100 мс.
		//Разблокировать прерывание 0.
		LD SM0.0
		MOVB 100, SMB34
		ATCH INT_0, 10
		ENI

INT0		Network 1 //Опросить аналоговый вход 4. LD SM0.0 MOVW AIW4,VW100

Основная программа

Эта основная часть программы содержит команды, управляющие вашим приложением. S7–200 выполняет эти команды последовательно и однократно в каждом цикле. Основная программа называется также OB1.

Подпрограммы

Эти необязательные элементы программы выполняются только тогда, когда они вызываются: основной программой, программой обработки прерываний или другой подпрограммой. Подпрограммы полезны, если вы хотите какую-нибудь функцию выполнять многократно. Чтобы не переписывать логику в каждом месте основной программы, где вы хотите выполнить эту функцию, вы можете записать логику функции один раз в подпрограмме, а затем вызывать эту подпрограмму столько раз, сколько необходимо при выполнении основной программы. Подпрограммы имеют много преимуществ:

- Использование подпрограмм уменьшает общую величину программы.
- Использование подпрограмм уменьшает время цикла, так как вы удалили соответствующий код из основной программы. S7–200 в каждом цикле анализирует код в основной программе независимо от того, исполняется этот код или нет, но код в подпрограмме анализируется только тогда, когда вы вызываете подпрограмму, и не анализируется в циклах, в которых подпрограмма не вызывается.
- С помощью подпрограмм создается мобильный код. Вы можете отграничить код для функции в подпрограмме, а затем копировать эту подпрограмму в другие программы без больших затрат.

Совет
Использование адресов памяти переменных может ограничить мобильность подпрограмм, так как назначение адресов в памяти переменных одной программы может привести к конфликту с назначением адресов в другой программе. Подпрограммы, которые используют для назначения всех адресов таблицу локальных переменных (локальные данные), напротив, хорошо переносятся, так как при применении локальных переменных невозможен конфликт адресов между подпрограммой и другими частями программы.

Программы обработки прерываний

Эти необязательные элементы программы реагируют на определенные прерывающие события. Программа обработки прерываний проектируется для обработки заранее определенных прерывающих событий. S7–200 исполняет программу обработки прерываний, когда возникает соответствующее событие.

Программы обработки прерываний не вызываются основной программой. Вы ставите программу обработки прерываний в соответствие прерывающему событию. S7–200 выполняет команды, находящиеся в программе обработки прерываний, только при возникновении прерывающего события.

Совет
Так как невозможно предсказать, когда S7–200 сгенерирует прерывание, то желательно ограничить количество переменных, используемых как в программе обработки прерываний, так и в других местах программы.

Используйте таблицу локальных переменных программы обработки прерываний, чтобы гарантировать, что ваша программа обработки прерываний будет использовать только временную память и не заменит данные, используемые где-нибудь еще в вашей программе.

Существует ряд методов программирования, которые гарантированно позволяют избежать ошибок при совместном использовании данных в основной программе и в программе обработки прерываний. Эти методы описаны в главе 6 вместе с командами прерываний.

Другие элементы программы

Другие блоки содержат информацию для S7–200. Вы можете загрузить эти блоки одновременно с загрузкой своей программы.


Системный блок

Системный блок

В системном блоке вы можете конфигурировать различные аппаратные возможности для S7–200.


Блок данных

Блок данных

Блок данных хранит значения для различных переменных (память переменных), используемых вашей программой. В блок данных можно вводить начальные значения для данных.

Использование STEP 7-Micro/WIN для создания вашей программы

Для открытия STEP 7-Micro/WIN дважды щелкните на символе STEP 7-Micro/WIN или выберите команду меню **Start > SIMATIC > STEP 7 MicroWIN 3.2 [Пуск > SIMATIC > STEP 7 MicroWIN 3.2]**. Как показано на рис. 5–1, окно проекта в STEP 7-Micro/WIN предоставляет удобную рабочую область для создания программы управления.

На панелях инструментов имеются кнопки для часто используемых команд меню. Вы можете эти панели инструментов по отдельности показывать или скрывать.

Навигационная панель предлагает группы символов для доступа к различным функциям программирования STEP 7-Micro/WIN.

Дерево команд отображает все объекты проекта и команды, необходимые для создания программы управления. Вы можете перетаскивать отдельные команды из этого дерева в свою программу или вставлять команду двойным щелчком в текущее положение курсора в редакторе программ.

Редактор программ содержит логику программы и таблицу локальных переменных, в которой вы можете назначить символические имена для временных локальных переменных.

Подпрограммы и программы обработки прерываний появляются как закладки в нижней части окна редактора программ. Для перемещения между подпрограммами, программами обработки прерываний и основной программой щелкайте по этим закладкам.

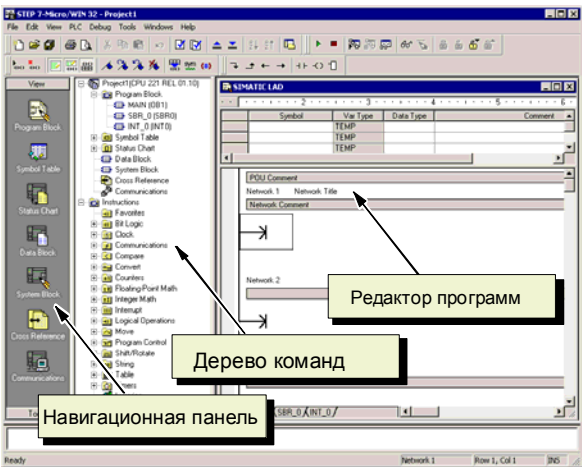


Рис. 5–1. STEP 7-Micro/WIN

Редактор программ

STEP 7-Micro/WIN имеет в своем распоряжении три редактора для создания программ: цепная логическая схема (LAD), , называемая также контактным планом (KOP), список команд (STL или AWL) и функциональная блок-схема (FBD), называемая также функциональным планом (FUP). С некоторыми ограничениями, программы, написанные в любом из этих редакторов программ, могут отображаться и редактироваться с помощью других редакторов программ.

Функции редактора STL

Редактор STL отображает программу на языке, имеющем текстовую основу. Редактор STL дает возможность создавать программы управления, вводя мнемонические обозначения команд. В редакторе STL можно создавать программы, которые невозможно создать в редакторе LAD или FBD. Это объясняется тем, что, используя STL, вы программируете на "родном языке" S7–200, а не в графическом редакторе, в котором имеются некоторые ограничения, чтобы можно было правильно изображать схемы соединений. Как показано на рис. 5–2, программирование в текстовом редакторе очень похоже на программирование на языках ассемблера.

S7–200 выполняет команды в порядке, определяемом программой, сверху вниз, а затем начинает сначала.

В STL логика управления реализуется с помощью логического стека. В STL вы должны вводить команды для обработки стековых операций.

```
LD   I0.0   //Прочитать вход
A    I0.1   //Выполнить логическое И-
            //сопряжение с другим входом
=    Q1.0   //Записать значение на выход 1
```

Рис. 5–2. Пример программы на STL

При выборе редактора STL примите во внимание следующее:

- STL лучше всего подходит опытным программистам.
- STL иногда позволяет решать проблемы, которые вы не можете достаточно легко решить при помощи редактора LAD или FBD.
- Вы можете использовать редактор STL только с системой команд SIMATIC.
- Тогда как вы всегда можете использовать редактор STL для просмотра или редактирования программы, созданной с помощью редактора LAD или FBD, обратное не всегда возможно. Вы не всегда можете использовать редактор LAD или FBD для отображения программы, написанной при помощи редактора STL.

Функции редактора LAD

Редактор цепных логических схем LAD отображает программу в виде графического представления, имеющего сходство с электрической монтажной схемой. Цепные логические схемы позволяют программе имитировать протекание электрического тока от источника напряжения через ряд логических условий на входах, которые, в свою очередь, активизируют логические условия на выходах. LAD-программа включает в себя находящуюся слева шину, находящуюся под напряжением, которая является источником потока сигнала. Замкнутые контакты позволяют потоку сигнала протекать через эти контакты к следующему элементу, а разомкнутые контакты препятствуют протеканию потока сигнала.

Логика подразделяется на сегменты. Программа исполняется сегмент за сегментом слева направо и сверху вниз. На рис. 5–3 показан пример программы в виде цепной логической схемы. Различные команды представляются графическими символами, имеющими три основные формы.

Контакты представляют логические состояния входов, например, выключателей, кнопок или внутренних условий.

Катушки обычно представляют логические результаты выходов, например, ламп, пускателей электродвигателей, промежуточных реле или внутренних выходных условий.

Блоки представляют дополнительные команды, например, таймеры, счетчики или математические команды.

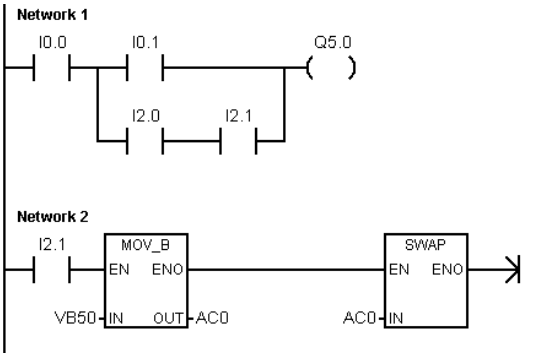


Рис. 5–3. Пример программы в виде цепной логической схемы

При выборе редактора LAD примите во внимание следующее:

- Цепная логическая схема проста в использовании для начинающих программистов.
- Графическое представление легко понимается и популярно во всем мире.
- Редактор LAD можно использовать и с системой команд SIMATIC, и с системой команд IEC 1131–3.
- Для отображения программы, созданной при помощи редактора SIMATIC LAD, всегда можно использовать редактор STL.

Функции редактора FBD

Редактор функционального плана FBD отображает программу в виде графического представления, напоминающего обычные логические схемы. Нет никаких контактов и катушек, как в редакторе LAD, но имеются эквивалентные команды, представленные в виде блоков.

На рис. 5–4 показан пример программы в виде функционального плана.

FBD не использует понятия левой и правой токовой шины; поэтому понятие «поток сигнала» выражает аналогичное понятие потока управления через логические блоки FBD.

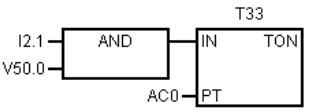


Рис. 5–4. Пример программы в виде функционального плана

По этой причине путь состояния «1» через элементы FBD называется потоком сигнала. Происхождение потока сигнала и место назначения его выхода ставятся в соответствие непосредственно операнду.

Логика программы вытекает из связей между функциональными блоками, обозначающими команды. Т.е. выход одной команды (например, блок логического И (AND)) может быть использован для разблокирования другой команды (например, таймера), формируя необходимую логику управления. Эта концепция позволяет решать широкий спектр задач управления.

При выборе редактора FBD примите во внимание следующее:

- Графическое представление функционального плана хорошо отражает процесс выполнения программы.
- Редактор FBD можно использовать и с системой команд SIMATIC, и с системой команд IEC 1131–3.
- Для отображения программы, созданной при помощи редактора SIMATIC FBD, всегда можно использовать редактор STL.

Наборы команд SIMATIC и IEC 1131-3

Большинство ПЛК предоставляют похожие основные команды, но обычно имеются незначительные различия в их внешнем виде, действии и т.д. в зависимости от поставщика. В течение последних нескольких лет Международная электротехническая комиссия (IEC) разработала всеобщий стандарт, который относится ко многим аспектам программирования ПЛК. Этот стандарт поощряет различных изготовителей ПЛК предлагать команды, являющиеся одинаковыми и по внешнему виду, и по действию.

Ваш S7–200 предлагает два набора команд, позволяющих решать широкий спектр задач автоматизации: набор команд IEC, соответствующий стандарту IEC 1131-3 для программирования ПЛК, и набор команд SIMATIC, разработанный специально для S7–200.

Совет
Когда STEP 7-Micro/WIN установлен в режим IEC, он отображает красный ромб (♦) в дереве команд рядом с командами, которые не определены стандартом IEC 1131-3.

Есть несколько ключевых различий между системой команд SIMATIC и системой команд IEC:

- Набор команд IEC ограничивается командами, которые являются стандартными среди поставщиков ПЛК. Некоторые команды, обычно включаемые в систему команд SIMATIC, не являются стандартными командами в спецификации IEC 1131–3. Они все еще доступны для использования как нестандартные команды, но если вы используете их, то программа больше не является строго совместимой с IEC 1131-3.
- У некоторых команд IEC в форме блоков возможна работа с несколькими форматами данных. Это свойство часто называют "перегрузкой". Например, вместо того, чтобы иметь отдельные математические блоки ADD_I (сложение целых чисел) и ADD_R (сложение вещественных чисел), команда ADD стандарта IEC 1131–3 проверяет формат складываемых данных и автоматически выбирает правильную команду S7–200. Это может несколько сэкономить затраты времени на программирование.
- Когда вы используете команды IEC, параметры команды автоматически проверяются на правильность формата данных, например, целое со знаком вместо целого без знака. Например, если вы попытались ввести целочисленное значение для команды, которая ожидала битовое значение (вкл/выкл), то происходит ошибка. Это свойство помогает минимизировать синтаксические ошибки программирования.

Делая выбор в пользу набора команд SIMATIC или IEC, примите во внимание следующие особенности:

- Команды SIMATIC обычно исполняются быстрее. Некоторые команды IEC могут иметь более длительные времена выполнения.
- Некоторые команды IEC, например, таймеры, счетчики, умножение и деление, работают иначе, чем их аналоги в SIMATIC.
- С набором команд SIMATIC можно использовать все три редактора программ (LAD, STL, FBD). С набором команд IEC можно использовать только редакторы LAD и FBD.
- Принцип действия команд IEC стандартизован для различных марок ПЛК, т.е. программы, удовлетворяющие IEC, могут разрабатываться независимо от системы автоматизации.
- Набор команд SIMATIC содержит больше операций, чем определено в стандарте IEC. Поэтому вы всегда можете включить команды SIMATIC в свою программу с командами IEC.
- IEC 1131-3 устанавливает, что переменные должны описываться с указанием типа, и поддерживает проверку типа данных системой.

Соглашения, используемые в редакторах программ

В STEP 7-Micro/WIN для всех редакторов действуют следующие соглашения:

- Знак # перед символическим именем (#var1) указывает, что этот символ имеет локальную сферу действия.
- Для команд IEC символ % указывает на прямую адресацию.
- Символ операнда «?.?» или «????»указывает, что требуется конфигурация операнда.

Программы, написанные в редакторе LAD, делятся на сегменты (network). Сегмент – это упорядоченное расположение контактов, катушек и блоков, которые соединены между собой, образуя замкнутую токовую цепь: при отсутствии коротких замыканий, разомкнутых цепей и условий для протекания потока сигнала в обратном направлении. STEP 7-Micro/WIN позволяет создавать комментарии к сегментам вашей программы, написанной в редакторе LAD. Программирование в редакторе FBD использует концепцию сегментов для разделения и комментирования вашей программы.

Программы на STL не используют сегментов; однако, вы можете использовать ключевое слово NETWORK для разбиения своей программы на части.

Соглашения, относящиеся к редактору LAD

В редакторе LAD вы можете использовать на своей клавиатуре клавиши F4, F6 и F9 для обращения к командам «Контакт», «Блок» и «Катушка». В редакторе LAD используются следующие соглашения:

- Символ «--->>»означает разомкнутую цепь или требование подключения потока сигнала.
- Символ «⇥»указывает, что выход представляет собой необязательный поток сигнала для команды, которая может быть включена каскадом или последовательно.
- Символ “>>” указывает, что вы можете использовать поток сигнала.

Соглашения, относящиеся к редактору FBD

В редакторе FBD вы можете использовать на своей клавиатуре клавиши F4, F6 и F9 для доступа к командам AND [И], OR [ИЛИ] и «Блок». Используются следующие соглашения:

- Символ “--->>” на операнде EN – это поток сигнала или индикатор операнда. Он может также изображать разомкнутую цепь или требование подключения потока сигнала.
- Символ «⇥»указывает, что выход представляет необязательный поток сигнала для команды, которая может быть включена каскадом или последовательно.
- Символы «<<» и «>>» показывают, что вы можете использовать значение или поток сигнала.
- Обозначение отрицания: Логическое отрицание NOT [НЕ] или инверсия состояния операнда или потока сигнала изображается небольшим кружком на входе. На рис. 5–5 Q0.0 равно результату логической операции НЕ I0.0 И I0.1. Такое обозначение отрицания действительно только для булевых сигналов, которые могут быть заданы как параметры или поток сигнала.

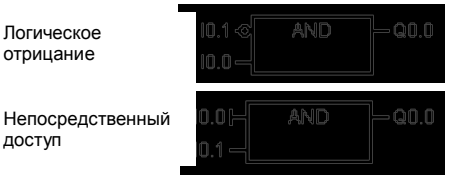


Рис. 5–5. Соглашения для FBD

- Индикаторы непосредственного управления: Как показано на рис. 5–5, редактор FBD изображает условие непосредственного управления булевым операндом вертикальной чертой на входе в команду FBD. Индикаторы непосредственного управления вызывают непосредственное чтение с указанного физического входа. Непосредственно управляемые операторы действительны только для физических входов.
- Блок без входов или выходов: Блок без входа обозначает команду, не зависимую от потока сигнала.

Совет
Количество операндов может быть расширено до 32 входов для команд AND [И] и OR [ИЛИ]. Для добавления и удаления обозначений операндов используйте клавиши «+» и «-» на своей клавиатуре.

Общие соглашения по программированию для S7–200

Определение EN/ENO

EN (Enable IN = Разрешающий вход) – это булев вход для блоков в LAD и FBD. Чтобы команда, представленная в виде блока, исполнялась, на этом входе должен присутствовать поток сигнала. В STL команды не имеют входа EN, но вершина стека должна быть логической “1”, чтобы соответствующая команда STL исполнялась.

ENO (Enable Out = Разрешающий выход) – это булев выход для блоков в LAD и FBD. Если у блока имеется поток сигнала на входе EN, и блок выполняет свою функцию без ошибок, то выход ENO передает поток сигнала следующему элементу. Если при исполнении блока обнаруживается ошибка, то поток сигнала завершается на блоке, в котором произошла ошибка.

В STL нет выхода ENO, но команды STL, соответствующие командам LAD и FBD с выходами ENO, устанавливают специальный бит ENO. Это бит доступен с помощью команды STL AENO (AND ENO) и может быть использован для создания того же эффекта, что и бит ENO блока.

Совет
Операнды EN/ENO и их типы данных в таблице действительных операндов для отдельных команд не показаны, так как эти операнды одинаковы для всех команд LAD и FBD. Таблица 5–1 перечисляет эти операнды и типы данных для LAD и FBD. Эти операнды применимы ко всем командам LAD и FBD, представленным в данном руководстве.

Таблица 5–1. Операнды и типы данных EN/ENO для LAD и FBD

Редактор программ	Входы/выходы	Операнды	Типы данных
LAD	EN, ENO	Поток сигнала	BOOL
FBD	EN, ENO	I, Q, V, M, SM, S, T, C, L	BOOL

Условные и безусловные входы

В LAD и FBD блок или катушка, зависящие от потока сигнала, изображаются присоединенными к какому-нибудь элементу с левой стороны. Катушка или блок, не зависящие от потока сигнала, изображаются непосредственно подключенными к левой шине. Условный и безусловный входы показаны в таблица 5–2.

Таблица 5–2. Представление условного и безусловного входов

Поток сигнала	LAD	FBD
Команда, зависящая от потока сигнала (условная)	1 —(JMP)	JMP
Команда, не зависящая от потока сигнала (безусловная)	—(NEXT)	NEXT

Команды без выходов

Блоки, не допускающие каскадного соединения, изображаются без булевых выходов. Сюда относятся команды вызова подпрограммы, перехода на метку и условного завершения подпрограммы. В LAD тоже имеются катушки, которые могут быть помещены только у левой шины. Сюда относятся также команды определения метки перехода, конца программного цикла с NEXT, загрузки реле последовательного управления (SCR), условного завершения SCR и конца SCR. В FBD они изображаются как блоки и отличаются непомеченными входами и отсутствием выходов.

Команды сравнения

Команда сравнения выполняется независимо от состояния потока сигнала. Если поток сигнала отсутствует (ложь), то выход ложен. Если поток сигнала присутствует (истина), то выход устанавливается в зависимости от результата сравнения.

Создание программы управления с помощью мастеров

В STEP 7-Micro/WIN имеются мастера, которые автоматизируют некоторые аспекты программирования и делают его более легким. В главе 6 команды, у которых есть соответствующий мастер, обозначены следующим символом:

Мастер команд

Устранение ошибок в S7–200

S7-200 разделяет ошибки на фатальные и не фатальные. Коды, сгенерированные ошибкой, можно посмотреть, выбрав команду меню **PLC > Information [ПЛК → Информация]**.

На рис. 5–6 показано диалоговое PLC Information [Информация ПЛК], содержащее и описание ошибки.

Поле Last Fatal [Последняя фатальная ошибка] показывает код предыдущей фатальной ошибки, сгенерированный S7–200. Это значение сохраняется при выключениях и включениях питания, если сохраняется ОЗУ. Эта ячейка очищается всякий раз, когда очищается вся память S7–200, или когда ОЗУ не сохраняется после длительного перерыва в подаче питания.

Поле Total Fatal [Всего фатальных ошибок] представляет собой количество фатальных ошибок, сформированных S7–200 начиная с момента последней очистки всех областей памяти S7–200. Это значение сохраняется при выключениях и включениях питания, если сохраняется ОЗУ. Эта ячейка очищается всякий раз, когда очищается вся память S7–200, или когда ОЗУ не сохраняется после длительного перерыва в подаче питания.

В Приложении C перечислены коды ошибок S7–200, а в Приложении D описаны биты специальной памяти (SM), которые могут быть использованы для контроля ошибок.

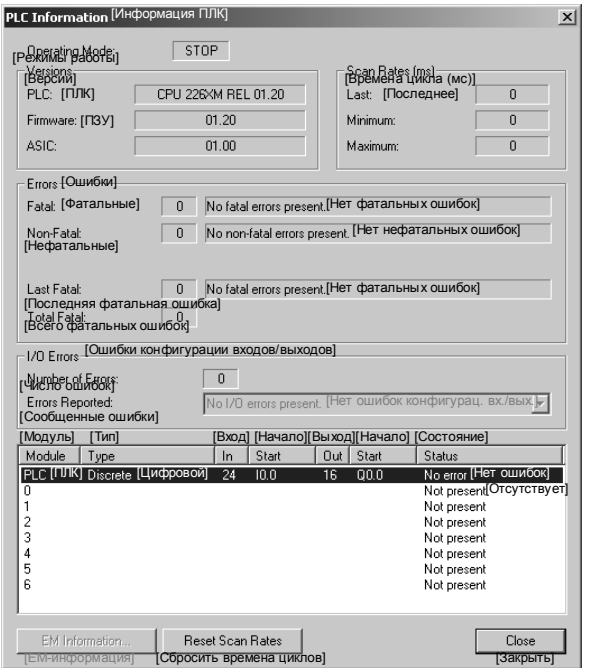


Рис. 5–6. Диалоговое окно с информацией ПЛК

Нефатальные ошибки

В случае нефатальных ошибок речь идет об ошибках в построении программы пользователя, об ошибке при исполнении команды в программе пользователя и об ошибках в модулях расширения. С помощью STEP 7-Micro/WIN можно отобразить коды нефатальных ошибок. Имеется три основных группы нефатальных ошибок.

Ошибки компиляции программы

S7–200 компилирует программу, когда он ее загружает. Если S7–200 обнаруживает, что программа нарушает правило компиляции, то загрузка прерывается и генерируется код ошибки. (Программа, которая уже была загружена в S7–200, по-прежнему будет существовать в ЭСППЗУ и не потеряется.) После исправления своей программы вы можете загрузить ее снова. Список нарушений правил компиляции вы найдете в Приложении C.

Ошибки конфигурации входов/выходов

При запуске S7–200 считывает конфигурацию входов-выходов из каждого модуля. При нормальной работе S7–200 периодически проверяет состояние каждого модуля и сравнивает его с конфигурацией, полученной при запуске. Если S7–200 обнаруживает разницу, он устанавливает бит ошибки конфигурации в регистре ошибок модуля. S7–200 не считывает входные данные из этого модуля и не записывает выходные данные в этот модуль, пока конфигурация модуля снова не совпадет с конфигурацией, полученной при запуске.

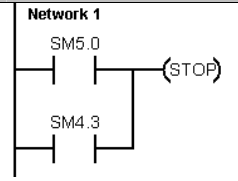
Информация о состоянии модуля хранится в битах специальной памяти (SM). Ваша программа может контролировать и анализировать эти биты. Подробную информацию о битах специальной памяти, используемых для индикации ошибок конфигурации входов/выходов вы найдете в Приложении D. Бит SM5.0 является глобальным битом ошибок конфигурации входов/выходов, который остается установленным, пока в модуле расширения сохраняется сбойная ситуация.

Ошибки выполнения программы

Ваша программа может создавать состояния ошибки во время своего выполнения. Эти ошибки могут возникать из-за ненадлежащего использования команды или из-за обработки командой недопустимых данных. Например, указатель косвенного адреса, который был действительным, когда программа компилировалась, может быть изменен во время выполнения программы так, что станет указывать на адрес вне допустимого диапазона. Это пример ошибки программирования, проявляющейся при выполнении программы. При возникновении такой ошибки устанавливается бит SM4.3. Он остается установленным, пока S7–200 находится в режиме RUN. (Список ошибок программирования, проявляющихся при выполнении программы, вы найдете в Приложении C). Информация об ошибках выполнения программы хранится в битах специальной памяти (SM). Ваша программа может контролировать и анализировать эти биты. Подробную информацию о битах специальной памяти, используемых для индикации ошибок исполнения программы, вы найдете в Приложении D.

Когда S7–200 обнаруживает нефатальную ошибку, он не переключается в режим STOP. Он только регистрирует событие в памяти SM и продолжает выполнение вашей программы. Однако вы можете спроектировать свою программу так, чтобы она принуждала S7–200 к переходу в состояние STOP, когда обнаруживается нефатальная ошибка. Следующий пример показывает сегмент программы, которая контролирует два глобальных бита нефатальных ошибок и переводит S7–200 в STOP всякий раз, когда устанавливается любой из этих битов.

Пример программы: Логика обнаружения нефатальной ошибки




```
Network 1   //При возникновении ошибки конфигурации входов/выходов
            //или ошибки выполнения программы перейти в STOP
LD      SM5.0
O       SM4.3
STOP
```

Фатальные ошибки

Фатальные ошибки заставляют S7–200 прекратить выполнение программы. В зависимости от тяжести фатальной ошибки S7–200 может потерять способность к выполнению некоторых или всех функций. Целью обработки фатальных ошибок является перевод S7–200 в безопасное состояние, из которого S7–200 может реагировать на запросы о существующих сбойных состояниях. Когда S7–200 обнаруживает фатальную ошибку, он переключается в режим STOP, включает светодиод System Fault [Ошибка системы] и светодиод STOP, заменяет таблицу выходов и выключает выходы. S7–200 остается в этом состоянии до исправления фатальной ошибки.

Как только вы сделали изменения для устранения фатальной ошибки, вы должны перезапустить S7–200, используя один из следующих методов:

- Выключение и затем включение питания.
- Перевод переключателя режимов работы из RUN или TERM в STOP.
- Выберите из STEP 7-Micro/WIN команду меню **PLC > Power–Up Reset [ПЛК > Сброс при запуске]** для запуска S7–200. Это заставляет S7–200 перезапуститься и сбросить все фатальные ошибки.

Перезапуск S7–200 сбрасывает состояние фатальной ошибки и выполняет диагностический тест, связанный с включением питания, чтобы проверить, что фатальная ошибка была устранена. Если обнаруживается другая фатальная ошибка, то S7–200 снова устанавливает светодиод ошибки, показывая, что ошибка по-прежнему существует. В противном случае S7–200 начинает нормальную работу.

Имеется несколько возможных сбойных состояний, которые могут сделать S7–200 неспособным к обмену данными. В этих случаях вы не можете отобразить код ошибки S7–200. Эти типы ошибок указывают на аппаратные отказы, требующие ремонта S7–200; их невозможно устранить посредством изменений в программе или очистки памяти S7–200.

Назначение адресов и начальных значений в редакторе блоков данных


Блок данных

В редакторе блоков данных вы можете выполнить присваивание начальных значений в памяти переменных. Вы можете выполнять назначения байтам, словам или двойным словам памяти переменных. Комментарии не обязательны.

Редактор блоков данных – это текстовый редактор со свободно выбираемым форматом; это значит, что поля для тех или иных данных заранее не определяются. После того как вы напечатали строку и нажали клавишу Enter, редактор блоков данных форматирует эту строку (выравнивает столбцы адресов, данных, комментариев; представляет адреса в памяти переменных большими буквами) и вновь ее отображает. Редактор блоков данных выделяет необходимое место в памяти переменных в соответствии со сделанными вами ранее назначениями адресов и размером (байт, слово или двойное слово) данных.

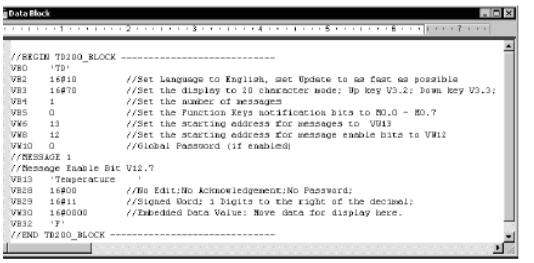


Рис. 5–7. Редактор блоков данных

Первая строка блока данных должна содержать явно назначенный адрес. Адреса в следующих строках могут быть назначены явно или неявно. Неявное присвоение адресов выполняется редактором, когда вы вводите несколько значений данных после назначенного адреса или вводите строку, содержащую только значения данных.

Редактор блоков данных принимает большие и маленькие буквы и допускает использование запятых, табуляций и пробелов в качестве разделителей между адресами и значениями данных.

Использование таблицы символов для символической адресации переменных


Таблица символов

Таблица символов дает возможность определять и редактировать символы, к которым можно обращаться во всей программе через символические имена. Вы можете создать несколько таблиц символов. В таблице символов имеется также закладка для символов, определенных системой, которые вы можете использовать в своей программе. Таблица символов называется также таблицей глобальных переменных.

Вы можете указывать операнды команд в своей программе абсолютно или символически. При абсолютной адресации задается область памяти, а также бит или байт адреса. При символической адресации для указания адреса используются комбинации алфавитно-цифровых символов.

Для программ SIMATIC назначение глобальных символов производится в таблице символов. Для программ IEC назначение глобальных символов производится в таблице глобальных переменных.

			Symbol	Address	Comment
1			AlwaysOn	SM0.0	Always on contact
2			Pump1	Q2.3	Pump 1 on/off
3			Pump1Limit	I1.1	Pump 1 pressure limit switch
4			Pump1Pressure	VD100	Pump 1 current pressure (real)
5			Pump1Rpm	VW200	Pump1 PRMs (integer)
6					

Рис. 5–8. Таблица символов

Для присвоения адресу символического имени:

1. Щелкните в навигационной панели на кнопке таблицы символов, чтобы вызвать таблицу.
2. Введите символическое имя (например, Input1) в столбце "Symbol Name [Символическое имя]". Максимальная длина символического имени составляет 23 символа.
3. В столбце Address [Адрес] введите адрес (например, I0.0).
4. Для таблицы глобальных переменных IEC введите значение в столбец Data Type [Тип данных] или выберите его из раскрывающегося окна списка.

Вы можете создать несколько таблиц символов, но вы не можете использовать одну и ту же строку более одного раза в качестве глобального символического имени ни в единственной таблице, ни в нескольких различных таблицах.

Использование локальных переменных

Таблицу локальных переменных редактора программ можно использовать для определения переменных, которые встречаются только в отдельной подпрограмме или программе обработки прерываний. См. рис. 5–9.

Локальные переменные могут использоваться как параметры, которые передаются в подпрограмму. Локальные переменные увеличивают мобильность и возможность повторного использования подпрограммы.

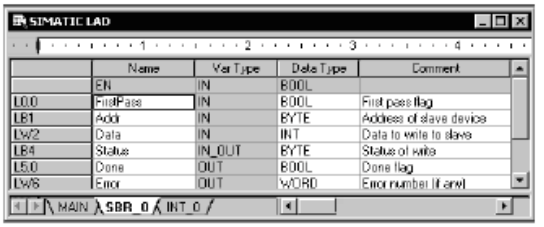


Рис. 5–9. Таблица локальных переменных

Контроль над программой с помощью таблицы состояний

Таблица состояний

С помощью таблицы состояний можно наблюдать и изменять переменные процесса, когда ваш S7–200 исполняет программу управления. Вы можете отслеживать состояние входов, выходов или переменных программы, отображая их текущие значения. В таблице состояний можно также принудительно задавать или изменять значения переменных процесса.

Вы можете создать несколько таблиц состояний, чтобы иметь возможность просматривать элементы из различных частей своей программы.

Для вызова таблицы состояний выберите команду меню **View > Component > Status Chart [Вид > Компонент > Таблица состояний]** или щелкните на пиктограмме таблицы состояний на навигационной панели.

При создании таблицы состояний введите адреса переменных процесса, которые вы хотите наблюдать. Невозможно отобразить состояния констант, аккумуляторов и локальных переменных. Значения таймеров и счетчиков можно отображать в виде бита или слова. Если значение отображается в виде бита, то оно представляет состояние бита таймера или счетчика; если значение отображается в виде слова, то оно является значением таймера или счетчика.

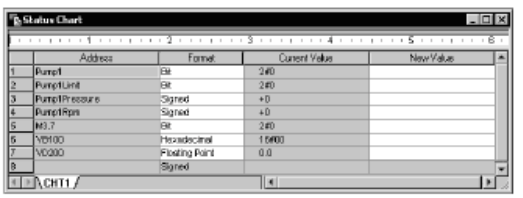


Рис. 5–10. Таблица состояний

Для создания таблицы состояний и контроля переменных:

1. Введите в поле адресов адреса желаемых величин.
2. В столбце Format выберите тип данных.
3. Для отображения состояния переменных процесса в своем S7–200 выберите команду меню **Debug > Chart Status [Отладка > Состояние таблицы]**.
4. Если вы хотите опрашивать эти величины непрерывно или хотите однократно считать состояние, щелкните на соответствующем символе на панели инструментов. В таблице состояний можно также принудительно устанавливать или изменять значения различных переменных процесса.

в таблицу состояний можно вставлять дополнительные строки, выбрав команду меню **Edit > Insert > Row [Редактировать > Вставить > Строка]**.

Совет
Вы можете создать несколько таблиц состояний, чтобы разделить перемененные на логические группы, чтобы каждую группу можно было наблюдать в собственно более короткой таблице.

Создание библиотеки команд

В STEP 7-Micro/WIN можно создать библиотеку команд для конкретного пользователя или использовать библиотеку, созданную другими лицами. См. рис. 5–11.

Для создания библиотеки команд создайте сначала в STEP 7-Micro/WIN подпрограммы и программы обработки прерываний и сгруппируйте их. Вы можете скрыть код в этих подпрограммах и программах обработки прерываний, чтобы предотвратить случайные изменения и защитить технологию или ноу-хау автора.

Для создания библиотеки команд действуйте следующим образом:

1. Напишите программу в виде стандартного проекта STEP 7-Micro/WIN и поместите функции, которые должны быть включены в библиотеку, в подпрограммы и программы обработки прерываний.
2. Обеспечьте, чтобы все адреса в памяти переменных в подпрограммах и программах обработки прерываний получили символические имена. В памяти переменных используйте адреса, следующие друг за другом, чтобы минимизировать размер памяти переменных, необходимой для библиотеки.
3. Переименуйте подпрограммы и программы обработки прерываний в соответствии с названиями, с которыми они должны находиться в библиотеке.
4. Выберите команду меню **File > Create Library [Файл > Создать библиотеку]**, чтобы скомпилировать новую библиотеку команд.

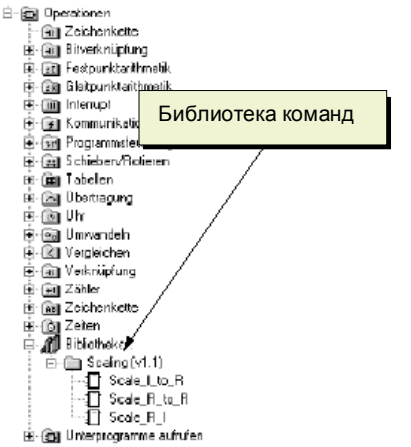


Рис. 5–11. Дерево команд с библиотеками

Дополнительную информацию о создании библиотек вы найдете в помощи для STEP 7-Micro/WIN в режиме online.

Чтобы получить доступ к командам из библиотеки команд, действуйте следующим образом:

1. Добавьте к дереву команд каталог "Libraries [Библиотеки]", выбрав команду меню **File > Add Libraries [Файл > Добавить библиотеки]**.
2. Выберите желаемую команду и вставьте ее в свою программу (как любую стандартную команду).

   Если библиотечная программа нуждается в памяти переменных, то STEP 7-Micro/WIN после компиляции проекта потребует назначения области памяти. Области памяти назначаются в диалоговом окне "Library Memory Allocation [Выделение памяти для библиотеки].

Функции тестирования программы

STEP 7-Micro/WIN предоставляет следующие функции для тестирования программы:

- Установка закладок в программе для облегчения поиска определенных строк программы.
- Таблица перекрестных ссылок, дающая возможность проверки ссылок, используемых в программе.
- Редактирование программы в режиме RUN, позволяющее выполнять небольшие изменения в пользовательской программе с минимальными помехами процессу, управляемому программой. При редактировании программы в режиме RUN вы можете также загрузить программный блок.

Дополнительную информацию о тестировании программы вы найдете в главе 8.