Основные концепции программирования CPU S7-200

4

Прежде чем вы начнете программировать ваше приложение, используя CPU S7-200, вы должны ознакомиться с некоторыми основными эксплуатационными характеристиками CPU.

Обзор главы

4.1 Рекомендации по проектированию системы с микроконтроллером

Есть много методов проектирования систем с микроконтроллером. Этот раздел предоставляет некоторые общие рекомендации, которые могут применяться к многим процессам проектирования. Конечно, вы должны следовать предписаниям процедур вашей собственной компании и технологий, принятых на основе вашего собственного обучения и местожительства. На рис. 4–1 показаны некоторые основные шаги процесса проектирования.

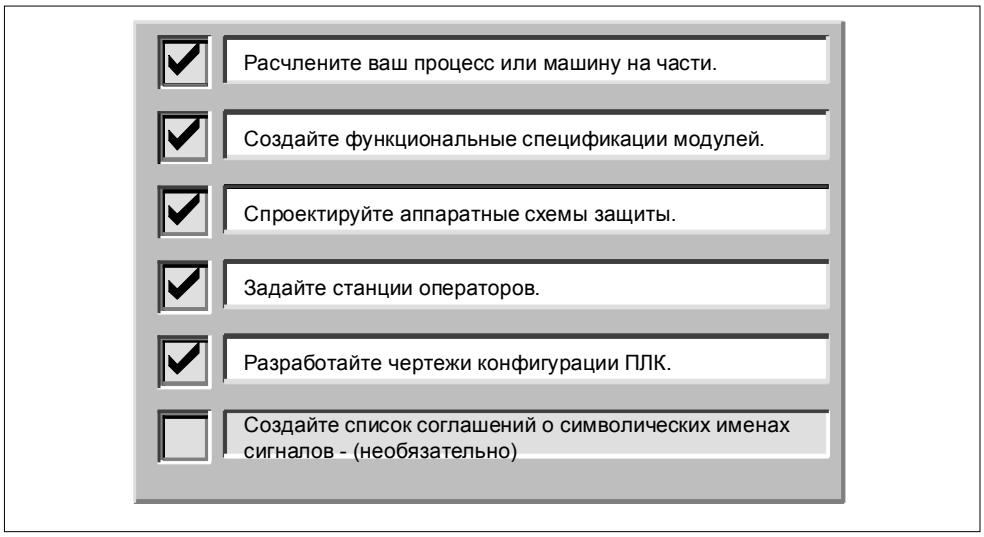


Рис. 4–1. Основные этапы проектирования системы с ПЛК

Расчленение процесса или машины

Разбейте ваш процесс или машину на сегменты, не зависящие друг от друга. Эти сегменты определяют границы между контроллерами и влияют на описания функциональных областей и назначение ресурсов.

Описание функциональных областей

Запишите описания работы каждого сегмента процесса или машины. Включите следующие пункты:

- Входы/выходы.
- Описание функционирования.
- Условия деблокировки (состояния, которые должны достигаться перед разрешением действия) для каждого исполнительного механизма (соленоиды, двигатели, приводы и т.д.).
- Описание интерфейса оператора.
- Интерфейсы с другими сегментами процесса или машины.

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 CPU, 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 CPU.
- Определение влияния CPU 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 CPU.
- Предоставление в CPU 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.

Задание 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 (дисплей, переключатели, лампы и т.д.).
- Электрические чертежи CPU 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 CPU относительно процесса или машины.
- Механическая компоновка CPU и модулей расширения входов-выходов (включая стойки и другое оборудование).
- Электрические чертежи для каждого CPU и модуля расширения входов-выходов (включая номера моделей устройств, коммуникационные адреса и адреса входов-выходов).

Разработка списка символических имен

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

4.2 Концепции программы S7-200

Связь программы с входами и выходами

Основная работа S7-200 CPU очень проста:

- CPU считывает состояние входов.
- Программа, хранимая в CPU, использует эти входы для вычисления логики управления. Во время выполнения программы CPU обновляет данные.
- CPU записывает данные в выходы.

Рис. 4–2 показывает простую схему, поясняющую, как связана электрическая релейная схема с CPU S7-200. В этом примере состояние переключателя на панели оператора для открытия стока добавляется к состояниям других входов. Затем вычисление этих состояний определяет состояние выхода, которое поступает в соленоид, закрывающий сток.

CPU циклически обрабатывает программу, считывая и записывая данные.

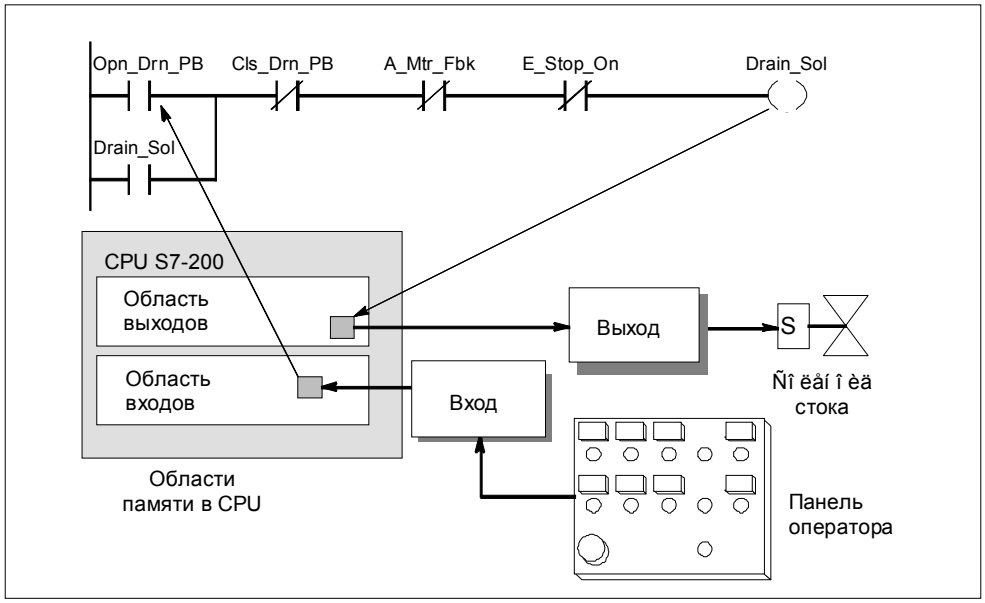


Рис. 4–2. Связь программы с входами и выходами

4.3 Концепции языков программирования и редакторов S7-200

CPU S7-200 предоставляют много типов команд, позволяющих вам решать широкий спектр задач автоматизации. В CPU S7-200 доступны две основные системы команд: SIMATIC и IEA 1131–3. Кроме того, наше ориентированное на ПК программное обеспечение для программирования STEP 7-Micro/WIN 32, предоставляет выбор различных редакторов, которые позволяют вам создавать управляющие программы с этими командами. Например, вы можете предпочитать создавать программы в среде, более ориентированной на графическое представление, тогда как кто-то другой в вашей компании может предпочитать в редакторе стиль языка ассемблера, основанный на текстовом представлении.

Всякий раз, когда вы разрабатываете ваши программы, у вас имеются для обдумывания два основных выбора:

- тип используемой системы команд (SIMATIC или IEA 1131-3)
- тип используемого редактора (Statement List [Список команд], Ladder Logic [Контактный план] или Function Block Diagram [Функциональный план]).

Возможны комбинации систем команд и редакторов S7-200, показанные в таблице 4–1.

Таблица 4–1. Системы команд и редакторов SIMATIC и IEA 1131–3

Система команд SIMATIC	Система команд IEA 1131–3
Редактор Statement List (STL)	STL недоступен
Редактор Ladder Logic (LAD)	Редактор Ladder Logic (LAD)
Редактор Function Block Diagram (FBD)	Редактор Function Block Diagram (FBD)

Редактор списка команд (Statement List)

Редактор списка команд (STL) STEP 7-Micro/WIN 32 позволяет создавать управляющие программы, вводя мнемонику команд. В общем случае редактор STL больше подходит для опытных программистов, хорошо знакомых с ПЛК и логическим программированием. Редактор STL позволяет также создавать программы, которые вы не могли бы создавать иным образом, используя редакторы для контактного или функционального плана. Поэтому вы программируете на родном языке CPU, а не в графическом редакторе, где должны применяться некоторые ограничения, чтобы правильно рисовать схемы. На рис. 4–3 показан пример программы в форме списка команд.

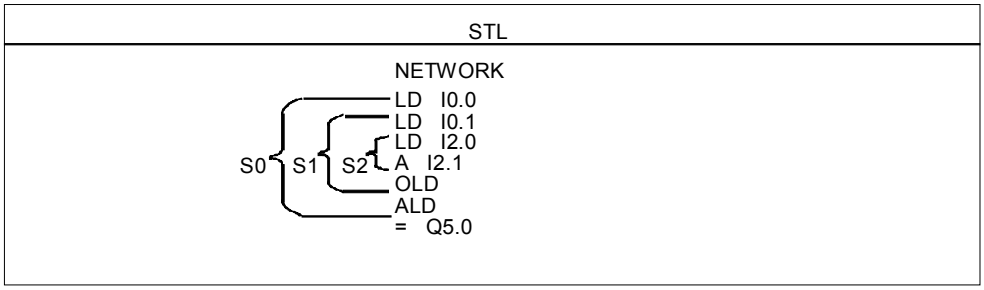


Рис. 4–3. Пример программы в форме STL

Как видно из рисунка 4–3, эта основанная на тексте концепция очень похожа на программирование на языке ассемблера. CPU выполняет команды в порядке, предписанном программой, сверху донизу и затем опять начинает сверху. STL и язык ассемблера похожи также в другом смысле. Для решения логики управления CPU S7-200 используют логический стек (см. рис. 4–4). Редакторы LAD и FBD вставляют команды, необходимые для обработки стека, автоматически. В STL эти команды для обработки стека должны вставляться вами.

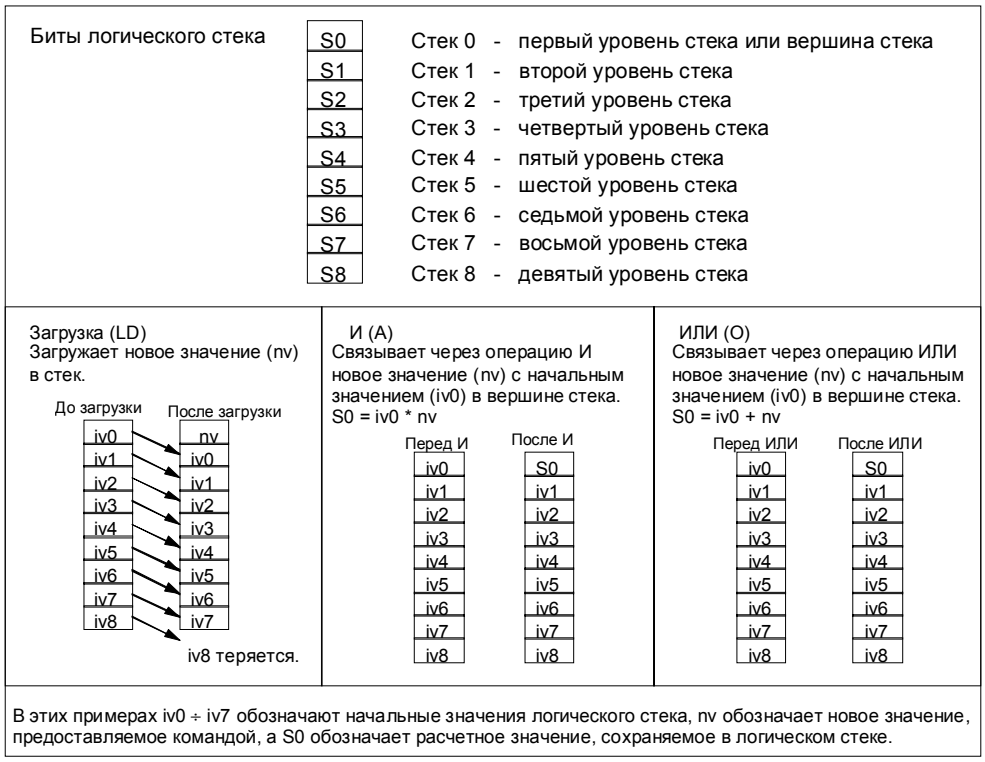


Рис. 4–4. Логический стек CPU S7-200

Основные особенности, принимаемые во внимание при выборе редактора STL:

- 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 SIMATIC LAD или FBD, обратное не всегда возможно. Вы не всегда можете использовать редактор SIMATIC LAD или FBD для отображения программы, написанной при помощи редактора STL.

Редактор контактного плана (Ladder Logic)

Редактор контактного плана (LAD) STEP 7-Micro/WIN 32 позволяет формировать программы, имеющие сходство с электрической монтажной схемой. Вероятно, программирование в контактном плане является методом, который выбирают многие программисты ПЛК и обслуживающий персонал. По существу, программы в контактном плане позволяют CPU эмулировать протекание электрического тока от источника питания через ряд логических состояний входов, которые, в свою очередь, разблокируют логические состояния выходов. Логика обычно подразделяется на малые легко понимаемые сегменты, которые в англоязычной литературе часто называются rungs, т.е. ступеньки (в соответствии с английским названием контактного плана – Ladder Logic, что буквально означает «лестничная логика»), или networks – цепи. Программа выполняется по “цепям” слева направо и затем сверху вниз, как предписано программой. Как только CPU достигает конца программы, он начинает снова с вершины программы.

Рис. 4–5 показывает пример программы в контактном плане.

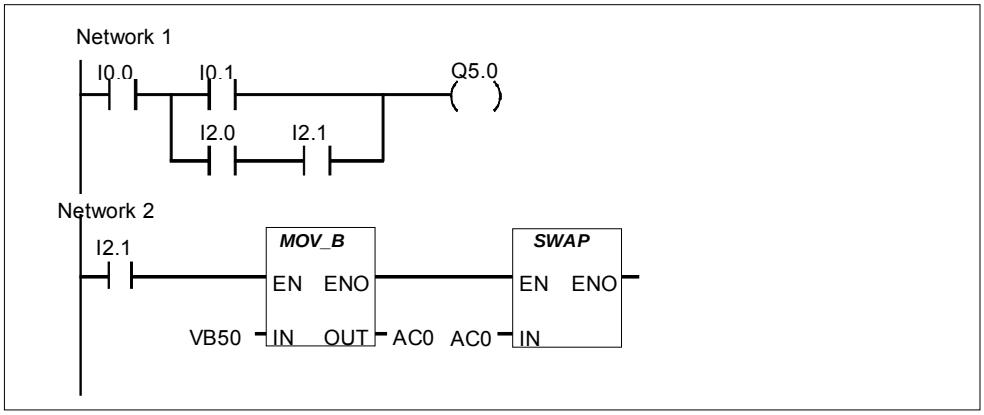


Рис. 4–5. Пример программы в форме LAD

Различные команды представляются графическими символами и включают три основные формы. Как показано на рисунке 4–5, вы можете последовательно соединять многочисленные команды в форме блоков.

- Контакты представляют логические состояния “входов”, аналогичных выключателям, кнопкам, внутренним условиям и так далее.
- Катушки обычно представляют логические результаты “выходов”, аналогичных лампам, пускателям электродвигателей, промежуточным реле, состояниям внутренних выходов и так далее.
- Блоки представляют дополнительные команды, такие как таймеры, счетчики или математические команды.

Основные особенности, принимаемые во внимание при выборе редактора LAD:

- Контактный план прост в использовании для начинающих программистов.
- Графическое представление часто более легко понимается и является популярным 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.

Редактор функционального плана (Function Block Diagram)

Редактор функционального плана (FBD) STEP 7-Micro/WIN 32 позволяет просматривать команды в форме логических блоков, напоминающих логические блок-схемы общего вида. Нет никаких контактов и катушек, как в редакторе LAD, но имеются эквивалентные команды, которые появляются как команды в форме блоков. Логика программы выводится из соединений между этими командами в форме блоков. То есть выход одной команды (например, блока AND) может использоваться для того, чтобы разрешить другой команде (например, таймеру) создать необходимую логику управления. Такая концепция соединений позволяет решать широкий круг логических задач.

На рис. 4–6 показан пример программы, созданной при помощи редактора функционального плана.

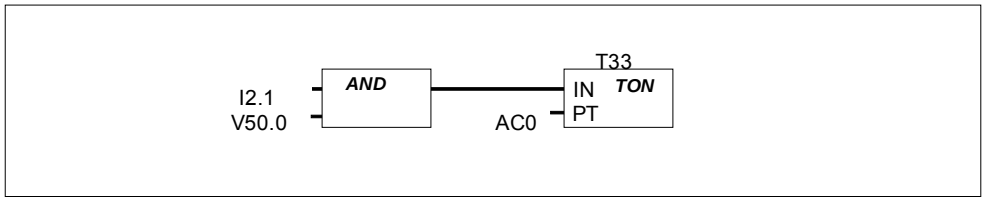


Рис. 4–6. Пример программы FBD

Основные особенности, принимаемые во внимание при выборе редактора FBD:

- Графический стиль представления в форме логических элементов хорош для последующего составления блок-схемы 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.

4.4 Объяснение различий между командами SIMATIC и IEC 1131–3

Система команд SIMATIC

Большинство ПЛК предоставляют похожие основные команды, но обычно имеются незначительные различия в их внешнем виде, действии и т.д. в зависимости от поставщика. Система команд SIMATIC предназначена для CPU S7-200. Эти команды могут выглядеть и работать немного по-другому, чем подобные команды в ПЛК с другим торговым знаком. Делая выбор в пользу системы команд SIMATIC, примите во внимание следующие особенности:

- Команды SIMATIC обычно имеют самые короткие времена выполнения.
- С системой команд SIMATIC работают все три редактора (LAD, STL, FBD).

Система команд IEC 1131-3

Международная электротехническая комиссия – это всемирное организация, разрабатывающая глобальные стандарты для всех областей электротехники. В течение последних нескольких лет эта комиссия разработала настоятельно необходимый стандарт, который точно соотносится со многими аспектами программирования ПЛК. Этот стандарт поощряет различных изготовителей ПЛК предлагать команды, являющиеся одинаковыми и по внешнему виду, и по действию. Есть несколько ключевых различий между системой команд SIMATIC и системой команд IEC1131-3.

- Система команд IEC 1131–3 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).
- Некоторые команды в форме блоков принимают множественные форматы данных. Эта технология часто упоминается как совмещение. 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 CPU. Это может сберечь ценное время при разработке программы.
- Когда вы используете команды IEC 1131–3, параметры команды автоматически проверяются на правильность формата данных. Проверка формата данных происходит неявно для пользователя. 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 команд IEC 1131–3, примите во внимание следующие особенности:

- Обычно проще изучать, как создаются программы для ПЛК различных торговых марок.
- В распоряжении имеется меньшее количество команд (как определено стандартом), но вы всегда можете использовать также многие из команд SIMATIC.
- Некоторые команды работают иначе, чем их аналоги в SIMATIC (таймеры, счетчики, умножение, деление и т.д.)
- Эти команды могут иметь более длительные времена выполнения.
- Эти команды можно использовать только в редакторах LAD и FBD.
- IEC 1131–3 устанавливает, что переменные должны описываться с указанием типа, и поддерживает проверку типа данных системой.

Типы данных для переменных в SIMATIC и IEC 1131-3

Каждая команда SIMATIC и IEC 1131-3 или параметризованная подпрограмма идентифицируется точным определением, к которому обращаются как к сигнатуре. Для всех стандартных команд допустимые типы данных каждого операнда команды получаются из сигнатуры. Для параметризованных подпрограмм сигнатура подпрограммы создается пользователем через таблицу локальных переменных.
STEP 7-Micro/WIN 32 осуществляет простую проверку типов данных для режима SIMATIC и строгую проверку типов данных для режима IEC 1131-3 режим. Когда тип данных определен и для локальной, и для глобальной переменной, STEP 7-Micro/WIN 32 гарантирует, что тип данных операндов соответствует сигнатуре команды для заданного уровня. Таблица 4–2 определяет элементарные типы данных, а таблица 4–3 показывает сложные типы данных, доступные в STEP 7-Micro/WIN 32.

Таблица 4–2. Элементарные типы данных IEC 1131–3

Элементарные типы данных	Размер типа данных	Описание	Диапазон данных
BOOL (1 бит)	1 бит	Булева переменная	0 ÷ 1
BYTE (8 битов)	8 битов	Байт без знака	0 ÷ 255
WORD (16 битов)	16 битов	Целое число без знака	0 ÷ 65 535
INT (16 битов)	16 битов	Целое число со знаком	-32768 ÷ +32767
DWORD (32 бита)	32 бита	Двойное целое число без знака	$0 \div 2^{32} - 1$
DINT (32 бита)	32 бита	Двойное целое число со знаком	$-2^{31} \div 2^{31} - 1$
REAL (32 бита)	32 бита	32-разрядное число с плавающей точкой IEEE	$-10^{38} \div +10^{38}$

Таблица 4–3. Сложные типы данных IEC 1131–3

Сложные типы данных	Описание	Диапазон адресов
TON	Таймер с задержкой включения	1 мс T32, T96 10 мс T33 ÷ T36, T97 ÷ T100 100 мс T37 ÷ T63, T101 ÷ T255
TOF	Таймер с задержкой выключения	1 мс T32, T96 10 мс T33 ÷ T36, T97 ÷ T100 100 мс T37 ÷ T63, T101 ÷ T255
TP	Импульсный таймер	1 мс T32, T96 10 мс T33 ÷ T36, T97 ÷ T100 100 мс T37 ÷ T63, T101 ÷ T255
CTU	Суммирующий счетчик	0 ÷ 255
CTD	Вычитающий счетчик	0 ÷ 255
CTUD	Реверсивный счетчик	0 ÷ 255
SR	Триггер с приоритетом установки	--
RS	Триггер с приоритетом сброса	--

Проверка типа данных: Существуют три уровня проверки типа данных: строгая проверка типа данных, простая проверка типа данных и отсутствие проверки типа данных.

Строгая проверка типа данных: В этом режиме тип данных параметра должен соответствовать типу данных символа или переменной. Каждый формальный параметр имеет только один тип данных (исключая совмещенные команды). Например, параметр IN в команде SRW (Shift Right Word [Сдвиг слова вправо]) имеет тип данных WORD. Только переменные, которым назначен тип данных WORD, будут компилироваться успешно. Переменные, которым назначен тип данных INT, недействительны для параметров типа WORD в командах, когда обеспечивается соблюдение строгой проверки типа данных.

Строгая проверка типа данных выполняется только в режимах IEC 1131-3. См. таблицу 4–4.

Таблица 4–4. Строгая проверка типа данных: выбранный пользователем и эквивалентный типы данных

Выбранный пользователем тип данных	Эквивалентный тип данных
BOOL	BOOL
BYTE	BYTE
WORD	WORD
INT	INT
DWORD	DWORD
DINT	DINT
REAL	REAL

Простая проверка типа данных: Если в режиме простой проверки типа данных символу или переменной присвоен тип данных, то им автоматически назначаются также все типы данных, которые соответствуют битовому размеру выбранного типа данных. Например, если вы выбираете тип данных DINT, то локальной переменной автоматически назначается также тип данных DWORD, потому что они оба являются 32-разрядными типами данных. Тип данных REAL автоматически не назначается, хотя он тоже является 32-разрядным типом данных. Тип данных REAL определен как не имеющий эквивалентного типа данных; он всегда уникален. Простая проверка типа данных выполняется только в режимах SIMATIC, когда вы используете локальные переменные. См. таблицу 4–5.

Таблица 4–5. Простая проверка типа данных: выбранный пользователем и эквивалентный типы данных

Выбранный пользователем тип данных	Эквивалентный тип данных
BOOL	BOOL
BYTE	BYTE
WORD	WORD, INT
INT	WORD, INT
DWORD	DWORD, DINT
DINT	DWORD, DINT
REAL	REAL

Отсутствие проверки типа данных: Режим отсутствия проверки типа данных доступен только для глобальных переменных SIMATIC, где типы данных не могут выбираться. В этом режиме символу автоматически назначаются все типы данных эквивалентного размера. Например, STEP 7-Micro/WIN 32 автоматически назначает символу, которому назначен адрес VD100, типы данных, показанные в таблице 4–6.

Таблица 4–6. Размер, определяемый типом данных для глобальных символов SIMATIC

Выбранный пользователем адрес	Назначенный эквивалентный тип данных
V0.0	BOOL
VB0	BYTE
VW0	WORD, INT
VD0	DWORD, DINT, REAL

Преимущества проверки типа данных

Проверка типа данных помогает вам избежать общих ошибок программирования. Если команда поддерживает числа со знаком, то STEP 7-Micro/WIN 32 отметит использование числа без знака для операнда команды. Например, операция сравнения < I является командой со знаком. В случае операндов, имеющих тип данных со знаком, число -1 меньше, чем 0. Однако, когда команде < I разрешается поддерживать тип данных без знака, программист должен гарантировать, что никогда не произойдет следующее. Во время выполнения программы для команды < I значение без знака 40 000 фактически меньше чем 0.

Предупреждение
Вы должны гарантировать, что число без знака, используемое в команде со знаком, не пересекает границу между положительными и отрицательными значениями. Отказ от обеспечения этого требования может порождать непредсказуемые результаты в вашей программе или в работе контроллера. Непредсказуемая операция контроллера может привести к смерти или серьезной травме персонала и/или существенному материальному ущербу. Всегда обеспечивайте, чтобы числа без знака для команд со знаком не пересекали границу между положительными и отрицательными значениями.

Итак, в режиме редактирования IEC 1131-3 строгая проверка типа помогает вам распознавать эти ошибки во время трансляции, вследствие генерирования ошибок для типов данных, не действительных для команды. Эта возможность недоступна для редакторов SIMATIC.

Выбор между режимами программирования SIMATIC и IEC 1131-3

Так как IEC 1131-3 определяет типы данных строго, а SIMATIC определяет типы данных не строго, STEP 7-Micro/WIN 32 не позволяет перемещать программы между этими двумя различными режимами редактирования. Вы должны выбрать предпочтительный режим редактирования.

Совмещенные команды

Совмещенные команды поддерживают ряд типов данных. Строгая проверка типа данных применяется по-прежнему, так как типы данных всех операндов должны быть согласованы прежде, чем команда скомпилируется успешно. Таблица 4–7 показывает пример совмещенной команды ADD стандарта IEC.

Таблица 4–7. Пример совмещенной команды ADD стандарта IEC

Команда	Разрешенные типы данных (строгая проверка типа данных)	Разрешенные типы данных (проверка типа данных)	Скомпилированная команда
ADD	INT	WORD, INT	ADD_I (сложение целых чисел)
ADD	DINT	DWORD, DINT	ADD_D (сложение двойных целых чисел)
ADD	REAL	REAL	ADD_R (сложение вещественных чисел)

Когда все операнды имеют тип данных DINT, компилятор генерирует команду «Сложить двойные целые числа». Если в совмещенной команде смешиваются типы данных, то происходит ошибка компиляции. Уровень проверки типа данных определяет то, что считается запрещенным. Следующий пример сгенерирует ошибку компилятора при строгой проверке типа данных, но пройдет компиляцию при простой проверке типа данных.

ADD IN1 = INT, IN2 = WORD, IN3 = INT

Строгая проверка типа данных: ошибка компиляции.
Проверка типа данных: компиляция в команду ADD_I (сложение целых чисел).

Простая проверка типа данных не предотвратит появление общих ошибок программирования во время выполнения. В случае простой проверки типа данных компилятор не будет улавливать следующие общие ошибки программирования: результат команды ADD 40000, 1 будет отрицательным числом, а не числом без знака 40001.

Использование прямой адресации в IEC для совмещенных команд

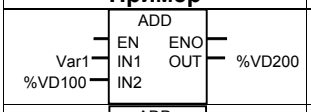
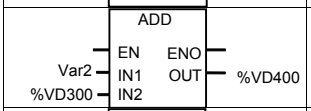
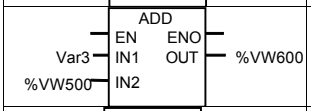
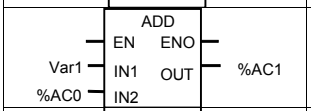
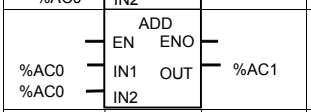
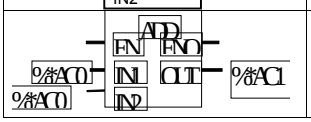
Режимы программирования IEC 1131-3 разрешают использовать непосредственно представленные ячейки памяти в качестве части набора параметров команды. В области параметров могут использоваться как переменные, так и ячейки памяти. Помните о том, что непосредственно представленные ячейки памяти не содержат в явном виде информацию о типе. Кроме того, информация о типе не может быть определена из любой совмещенной команды IEC, потому что эти команды принимают меняющиеся типы данных.

Типы данных непосредственно представленных параметров определяются путем исследования других параметров с заданным типом, включенных в состав команды. Когда тип параметров команды сконфигурирован так, чтобы использовать переменную, имеющую конкретный тип, все непосредственно представленные параметры будут предполагаться имеющими такой же тип. Таблицы 4–8 и 4–9 показывают примеры типов данных непосредственно представленных параметров. Для обозначения прямого адреса используется знак процента (%).

Таблица 4–8. Пример типов данных для прямой адресации

Имя	Адрес	Тип данных	Комментарий
Var1		REAL	Это переменная с плавающей точкой.
Var2		DINT	Это переменная для двойного целого числа.
Var3		INT	Это переменная для целого числа.

Таблица 4–9. Примеры прямой адресации в совмещенных командах

Пример	Описание
	VD100 и VD200 будут предполагаться имеющими тип REAL, так как Var1 имеет тип REAL.
	VD300 и VD400 будут предполагаться имеющими тип DINT, так как Var2 имеет тип DINT.
	VW500 и VW600 будут предполагаться имеющими тип INT, так как Var3 имеет тип INT.
	AC0 и AC1будут предполагаться имеющими тип REAL, так как Var1 имеет тип REAL.
	Эта конфигурация запрещена, так как тип не может быть определен. Тип данных в аккумуляторах может быть любым.
	Эта конфигурация запрещена, так как тип не может быть определен. Тип данных в указателях аккумуляторов может быть любым.

% Обозначает прямой адрес.

Использование команд преобразования

Команды преобразования позволяют перейти от одного типа данных к другому. STEP 7-Micro/WIN 32 поддерживает команды преобразования, показанные в таблице 4–10, для перемещения значений между простыми типами данных.

Таблица 4–10. Команды преобразования

Команда преобразования	Строгая проверка типа данных Разрешенные операнды	Проверка типа данных Разрешенные операнды
BYTE → INT	IN: BYTE OUT: INT	IN: BYTE OUT: WORD, INT
INT → BYTE	IN: INT OUT: BYTE	IN: WORD, INT OUT: BYTE
INT → DINT	IN: DINT OUT DINT	IN: WORD, INT OUT: DWORD, DINT
DINT → INT	IN: DINT OUT: INT	IN: DWORD, DINT OUT: WORD, INT
DINT → REAL	IN: DINT OUT: REAL	IN: DWORD, DINT OUT: REAL
REAL → DINT (ROUND)	IN: REAL OUT: DINT	IN: REAL OUT: DWORD, DINT

В режиме редактирования IEC 1131-3 вы можете использовать для преобразований между INT и WORD и между DINT и DWORD совмещенную команду MOVE. Команда MOVE разрешает для перемещаемых операндов типы данных одинакового размера без генерирования компилятором ошибок. См. таблицу 4–11.

Таблица 4–11. Использование совмещенной команды MOVE

Совмещенная MOVE в IEC 1131-3	IN	OUT
MOVE (INT → WORD)	INT	WORD
MOVE (WORD → INT)	WORD	INT
MOVE (DINT → DWORD)	DINT	DWORD
MOVE (DWORD → DINT)	DWORD	DINT

4.5 Основные элементы для построения программы

CPU S7-200 непрерывно выполняет вашу программу, чтобы управлять задачей или процессом. Вы создаете эту программу при помощи STEP 7-Micro/WIN 32 и загружаете ее в CPU. Вы можете из главной программы вызывать различные подпрограммы или программы обработки прерываний.

Организация программы

Программы для CPU S7-200 конструируются из трех основных элементов: главная программа, (необязательные) подпрограммы и (необязательные) подпрограммы обработки прерываний. Программа S7-200 делится на следующие организационные элементы:

- Главная программа: Основная часть программы, где размещаются команды, управляющие вашим приложением. Команды главной программы выполняются последовательно и однократно в каждом цикле сканирования CPU.
- Программы обработки прерываний: Эти необязательные элементы программы выполняются при каждом возникновении события, вызывающего прерывание.
- Подпрограммы: Эти необязательные элементы программы выполняются только тогда, когда они вызываются из главной программы или программы обработки прерываний.

Пример программы, использующей подпрограммы и прерывания

Ниже следуют примеры программ для циклического прерывания, которое может использоваться в таких приложениях, как считывание значения аналогового входа. В этом примере типовой интервал времени опроса аналогового входа задан равным 100 мс.

На рисунках 4–7 ÷ 4–11 показаны программы, использующие подпрограмму и программу обработки прерываний для различных языков программирования S7-200.

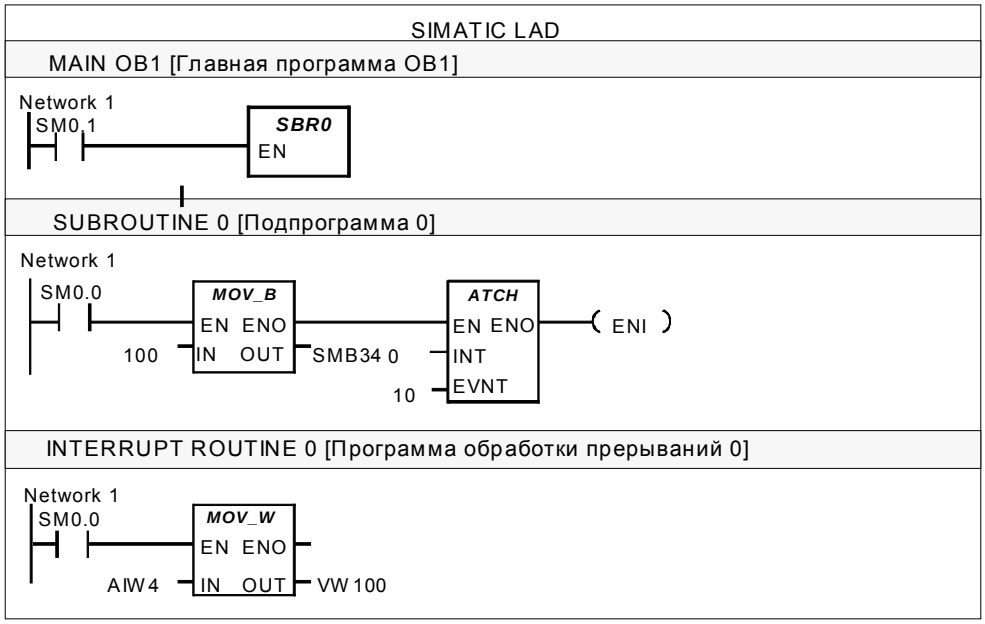


Рис. 4–7. Программа SIMATIC LAD с использованием подпрограммы и программы
обработки прерываний

Список команд

Main Program OB1 [Главная программа OB1]

```
Network 1
LD    SM0.1   // Когда включается бит первого цикла обработки программы,
CALL    0   // вызвать подпрограмму 0
```

Subroutine 0 [Подпрограмма 0]

```
Network 1
LD   SM0.0   // Бит памяти, включенный всегда
MOVB   100, SMB34   // Установить интервал циклического прерывания 0
                                  // на 100 мс
AENO        // Если пересылка успешна, то
ATCH   0, 10    // назначить циклическое прерывание 0
                // программе обработки прерываний 0
AENO      // Если назначение прошло успешно, то
ENI       // разрешить глобальное прерывание
```

Interrupt Routine 0 [Программа обработки прерываний 0]

```
    // Начало программы обработки прерываний 0
Network 1
LD SM0.0   // Бит памяти, включенный всегда
MOVW   AIW4,VW100   //  Взять отсчет аналогового входа 4
```

Рис. 4–8. Программа SIMATIC STL с использованием подпрограммы и программы обработки прерываний

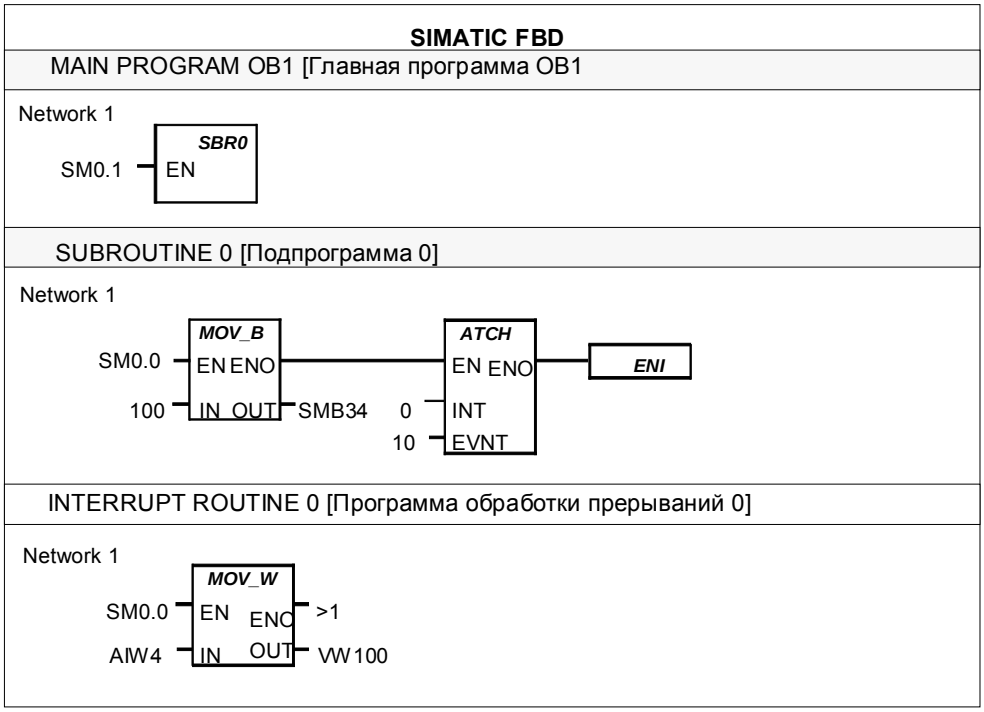


Рис. 4–9. Программа SIMATIC FBD с использованием подпрограммы и программы обработки прерываний

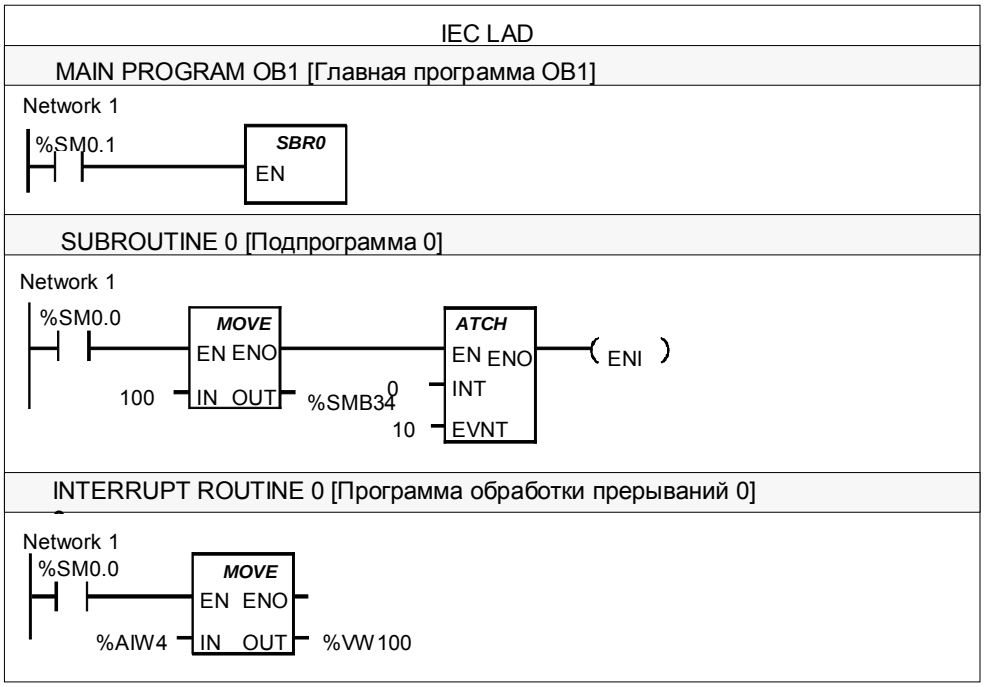


Рис. 4–10. Программа IEC LAD с использованием подпрограммы и программы обработки прерываний

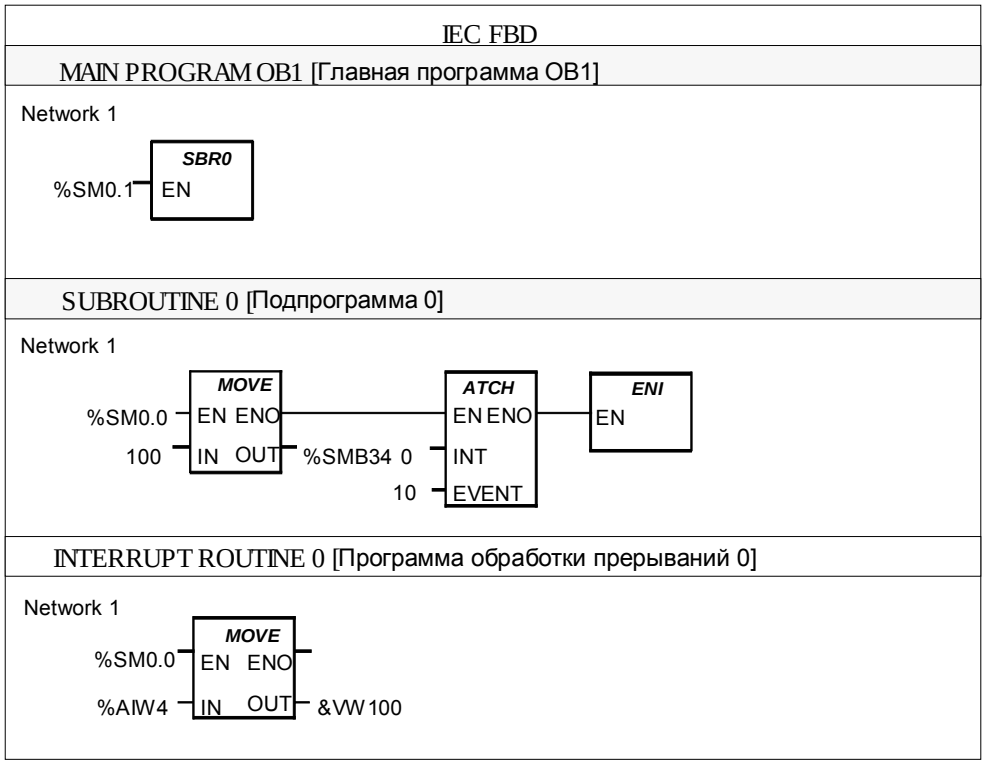


Рис. 4–11. Программа IEC FBD с использованием подпрограммы и программы обработки прерываний

4.6 Объяснение цикла сканирования CPU

CPU S7-200 предназначен для циклического выполнения ряда заданий, включая вашу программу. Такое циклическое выполнение заданий называется циклом сканирования. В течение цикла сканирования, показанного на рисунке 4–12, CPU выполняет все или большинство из следующих задач:

- считывание входов
- выполнение программы
- обработка любых коммуникационных запросов
- выполнение самодиагностики CPU
- запись в выходы.

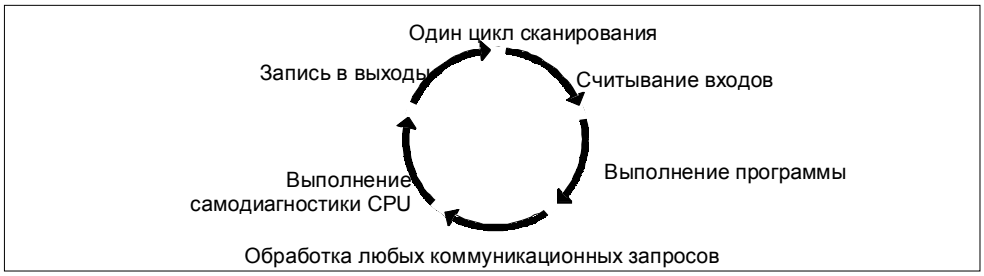


Рис. 4–12. Цикл сканирования CPU S7-200

Ряд заданий, выполняемых в течение цикла сканирования, зависит от режима работы CPU. CPU S7-200 имеет два режима работы – режим STOP и режим RUN. С точки зрения цикла сканирования основное различие между режимами STOP и RUN состоит в том, что в режиме RUN программа выполняется, а в режиме STOP программа не выполняется.

Считывание цифровых входов

Каждый цикл сканирования начинается со считывания текущих значений цифровых входов и последующей записи этих значений в регистр входов образа процесса.

CPU резервирует регистр входов образа процесса приращениями по восемь битов (один байт). Если CPU или модуль расширения не предоставляет физическую точку ввода для каждого бита зарезервированного байта, то вы не можете перераспределить эти биты последующим модулям в цепи ввода-вывода или использовать их в вашей программе. CPU обнуляет эти неиспользованные входы в регистре образа процесса в начале каждого цикла сканирования. Однако, если ваш CPU может принять несколько модулей расширения и вы не используете эту возможность ввода-вывода (не установили модули расширения), то вы можете применить неиспользованные биты входов расширения как дополнительные биты памяти.

CPU не обновляет аналоговые входы во время нормального цикла сканирования, если не разрешена цифровая фильтрация аналоговых входов. Цифровая фильтрация предоставляется как выбираемая пользователем возможность и может индивидуально разрешаться для каждого аналогового входа.

Цифровая фильтрация предназначена для использования с дешевыми аналоговыми модулями, которые не обеспечивают фильтрацию внутри модуля. Цифровая фильтрация должна использоваться в приложениях, где входной сигнал с течением времени изменяется медленно. Если сигнал изменяется с высокой скоростью, то цифровая фильтрация не должна разрешаться.

Когда фильтрация аналогового входного сигнала для аналогового входа разрешена, CPU один раз за цикл сканирования обновляет этот аналоговый вход, выполняет функцию фильтрации и сохраняет фильтрованное значение внутри. Затем фильтрованное значение предоставляется каждый раз, когда ваша программа обращается к аналоговому входу.

Если аналоговая фильтрация для аналогового входа не разрешена, то CPU считывает значение аналогового входа из физического модуля каждый раз, когда ваша программа обращается к аналоговому входу.

Выполнение программы

В фазе выполнения цикла сканирования CPU выполняет программу, начиная с первой команды и продолжая до конечной команды. Команды непосредственного ввода-вывода дают немедленный доступ к входам и выходам во время выполнения программы или программы обработки прерывания.

Если вы используете в своей программе прерывания, то программы обработки прерываний, связанные с событиями, вызывающими прерывание, хранятся как часть программы (см. раздел 4.5). Программы обработки прерываний не выполняются в качестве части нормального цикла сканирования, а выполняются, когда происходит прерывающее событие (которое может возникнуть в любой точке цикла сканирования).

Обработка коммуникационных запросов

Во время фазы обработки сообщений в цикле сканирования CPU обрабатывает любые сообщения, принятые из коммуникационного порта.

Выполнение самодиагностики CPU

Во время этой фазы цикла сканирования CPU проверяет свое встроенное программное обеспечение и память вашей программы (только режим RUN). Он проверяет также состояние всех модулей ввода-вывода.

Запись в цифровые выходы

В конце каждого цикла сканирования CPU записывает значения, хранимые в регистре выходов образа процесса, в цифровые выходы.

CPU резервирует регистр выходов образа процесса приращениями по восемь битов (один байт). Если CPU или модуль расширения не предоставляет физическую точку вывода для каждого бита зарезервированного байта, то вы не можете перераспределить эти биты последующим модулям в цепи ввода-вывода.

Когда режим работы CPU изменяется с RUN на STOP, в цифровых выходах устанавливаются значения, определенные в таблице выходов, или сохраняются их текущие состояния (см. раздел 6.4). Заданное по умолчанию действие состоит в том, что цифровые выходы выключаются. Аналоговые выходы остаются с последними записанными значениями.

Прерывание цикла сканирования

Если вы используете прерывания, то программы, связанные с каждым событием, вызывающим прерывание, хранятся как часть программы. Программы обработки прерываний не выполняются как часть нормального цикла сканирования, f выполняются, когда происходит прерывающее событие (которое может возникнуть в любой точке цикла сканирования). Прерывания обслуживаются CPU по принципу “первым поступил – первым обслужен” в пределах назначенных им приоритетов.

Входной и выходной регистры образа процесса

Во время выполнения программы обычно выгодней использовать регистр образа процесса, чем обращаться непосредственно к входам или выходам. Для использования регистров образа процесса есть три причины:

- Выборка значений всех входов в начале сканирования синхронизирует и “замораживает” значения входов на время фазы выполнения программы в цикле сканирования. Выходы обновляются из регистра образа процесса после того, как выполнение программы закончено. Это оказывает стабилизирующее воздействие на систему.
- Ваша программа может обращаться к регистру образа процесса намного быстрее, чем к входам-выходам, давая возможность более быстрого выполнения программы.
- Входы и выходы – это битовые объекты, и к ним нужно обращаться как битам, но к регистру образа процесса вы можете обращаться как к битам, байтам, словам или двойным словам. Таким образом, регистры образа процесса предоставляют дополнительную гибкость.

Непосредственный ввод–вывод

Команды непосредственного ввода–вывода дают возможность прямого доступа к фактической точке ввода или вывода, хотя обычно при обращениях к входам–выходам в качестве источника или адресата используются регистры образа процесса. Когда вы используете команду непосредственного обращения к входу, соответствующая ячейка регистра входов образа процесса не изменяется. Когда вы используете команду непосредственного обращения к выходу, одновременно обновляется соответствующая ячейка регистра выходов образа процесса.

CPU обрабатывает аналоговый вход–выход как непосредственные данные, если цифровая фильтрация аналогового входа не разрешена. См. раздел 6.5. Когда вы записываете значение в аналоговый выход, этот выход немедленно обновляется.

4.7 Выбор режима работы CPU

CPU 7-200 имеет два режима работы:

- STOP: CPU не выполняет программу. Когда CPU находится в режиме STOP, вы можете загружать программу или конфигурировать CPU.
- RUN: CPU выполняет программу.

Светодиод состояния на лицевой стороне CPU показывает текущий режим работы.

Вы можете изменить режим работы:

- вручную переключая переключатель режима, расположенный на CPU;
- используя программное обеспечение для программирования STEP 7-Micro/WIN 32 и устанавливая переключатель режима CPU в положение TERM или RUN;
- вставив команду STOP в вашу программу.

Изменение режима работы при помощи переключателя режима

Вы можете использовать переключатель режима (расположенный под передней дверцей CPU), чтобы вручную выбрать режим работы CPU:

- Установка переключателя режима в положение STOP останавливает выполнение программы.
- Установка переключателя режима в положение RUN запускает выполнение программы.
- Установка переключателя режима в положение TERM (terminal), не изменяет режима работы CPU.

Если выключение и последующее включение питания происходит, когда переключатель режима установлен в положение STOP или TERM, то при восстановлении питания CPU автоматически переходит в режим STOP.

Если выключение и последующее включение питания происходит, когда переключатель режима установлен в положение RUN, то при восстановлении питания CPU переходит в режим RUN.

Изменение режима работы при помощи STEP 7-Micro/WIN 32

Как показано на рисунке 4–13, для изменения режима работы CPU вы можете использовать STEP 7-Micro/WIN 32. Чтобы дать возможность программному обеспечению изменить режим работы, вы должны установить переключатель режима CPU в положение TERM или RUN.

Рис. 4–13. Использование STEP 7-Micro/WIN 32 для изменения режима работы CPU

Пояснения к рисунку: Project – проект; Edit – редактирование; View – вид; PLC – ПЛК; Debug – отладка; Tools – инструментальные средства; Windows – окна.

Изменение режима работы из программы

Вы можете вставить в вашу программу команду STOP, чтобы переключить CPU в режим STOP. Это позволяет остановить выполнение вашей программы на основе ее логики. Для получения дополнительной информации о команде STOP см. главу 9 для команд SIMATIC и главу 10 для команд IEC 1131-3.

4.8 Создание пароля для CPU

Все модели CPU S7-200 обеспечивают парольную защиту для ограничения доступа к определенным функциям CPU. Пароль санкционирует доступ к функциям CPU и памяти. При отсутствии пароля CPU предоставляет неограниченный доступ. При наличии парольной защиты CPU запрещает все ограниченные операции в соответствии с обеспечиваемой конфигурацией, когда пароль был установлен.

Ограничение доступа к CPU

Как показано в таблице 4–12, CPU S7-200 предоставляют три уровня ограничения доступа к функциям CPU. Каждый уровень позволяет обращаться к определенным функциям без пароля. Для всех трех уровней доступа ввод правильного пароля предоставляет доступ ко всем функциям CPU. Для CPU S7-200 по умолчанию задан уровень 1 (нет ограничения доступа).

Ввод пароля через сеть не ставит под угрозу парольную защиту CPU. Наличие одного пользователя, уполномоченного обращаться к функциям CPU ограниченного доступа, не разрешает другим пользователям обращаться к этим функциям. В конкретный момент времени только одному пользователю разрешается неограниченный доступ к CPU.

Примечание
После того, как вы введете пароль, уровень прав доступа для этого пароля остается действительным в течение одной минуты с момента отсоединения устройства программирования от CPU. Если в течение этого времени другой пользователь немедленно соединится с CPU, то он может иметь доступ к устройству программирования.

Таблица 4–12. Ограничение доступа к CPU S7–200

Задача	Уровень 1	Уровень 2	Уровень 3
Чтение и запись данных пользователя	Не ограничено	Не ограничено	Не ограничено
Запуск, останов и перезапуск CPU			
Чтение и запись часов реального времени			
Загрузка программы пользователя, данных и конфигурации			Требуется пароль
Загрузка в CPU		Требуется пароль	
Состояние STL			
Стирание программы пользователя, данных и конфигурации			
Принудительное задание данных или однократного/многократного сканирования			
Копирование в модуль памяти			
Запись выходов в режиме STOP			

Создание пароля CPU

Чтобы создать пароль для CPU, вы можете использовать STEP 7-Micro/WIN 32 . Выберите команду меню **View → System Block [Вид → Системный блок]** и щелкните по вкладке Password [Пароль]. См. рис. 4–14. Введите соответствующий уровень прав для CPU, затем введите и проверьте пароль для CPU.

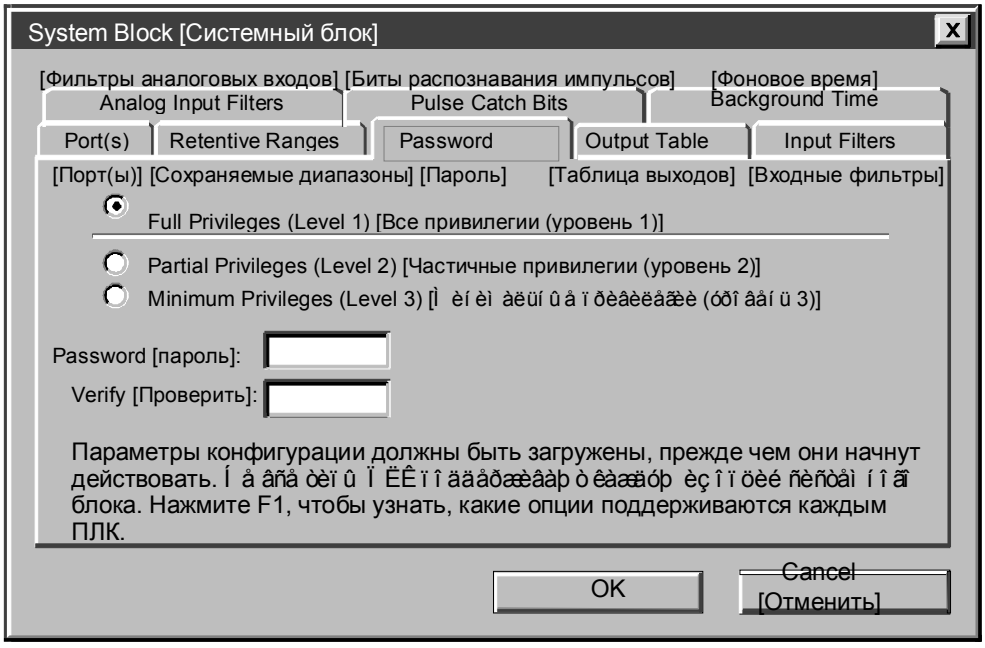


Рис. 4–14. Создание пароля для CPU

Что делать, если вы забыли пароль для CPU

Если вы забыли пароль для CPU, то вы должны очистить память CPU и перезагрузить вашу программу. Очистка памяти CPU переводит CPU в режим STOP и устанавливает в CPU значения заводской настройки, за исключением адреса CPU, скорости передачи и часов реального времени. Для стирания программы в CPU выберите команду меню **PLC → Clear... [ПЛК → Очистить...]**, чтобы отобразить диалоговое окно Clear [Очистка]. Выделите все три блока и подтвердите ваше действие щелчком на кнопке OK. Если пароль создан, то отображается диалоговое окно, в котором запрашивается пароль доступа. Ввод пароля clearplc позволяет вам продолжить операцию Clear All [Очистить все].

Операция Clear All [Очистить все] не удаляет программу из модуля памяти. Так как модуль памяти наряду с программой хранит пароль, вы должны перепрограммировать также модуль памяти, чтобы удалить потерянный пароль.

Предупреждение
Очистка памяти CPU вызывает выключение выходов (или “замораживание” на определенном уровне в случае аналогового выхода). Если во время очистки памяти CPU S7-200 соединен с оборудованием, то изменения состояний выходов могут передаваться оборудованию. Если вы конфигурировали для выходов “безопасное состояние”, отличающееся от заводской настройки, то изменения выходов могут вызвать непредсказуемую реакцию вашего оборудования, которая может также вызвать смерть или серьезную травму персонала и/или повреждение оборудования. Всегда соблюдайте соответствующие предосторожности по безопасности и перед очисткой памяти CPU обеспечьте, что ваш процесс находится в безопасном состоянии.

4.9 Отладка и контроль вашей программы

STEP 7-Micro/WIN 32 предоставляет множество инструментов для отладки и контроля вашей программы

Использование первого или многократного сканирования для контроля вашей программы

Вы можете задать, чтобы CPU выполнял вашу программу в течение ограниченного числа сканирований (от 1 до 65535 сканирований). Выбирая число сканирований для прогона CPU, вы можете наблюдать, как программа изменяет переменные процесса. Чтобы задать количество выполняемых сканирований, используйте команду меню **Debug → Multiple Scans [Отладка → Многократные сканирования]**. Рис. 4–15 показывает диалоговое окно ввода числа выполняемых сканирований CPU.

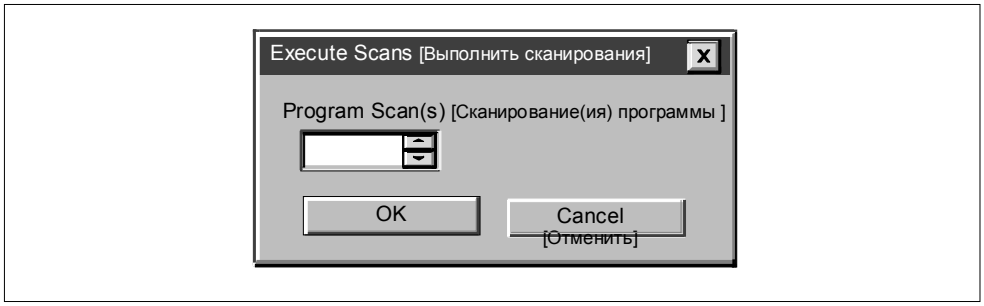


Рис. 4–15. Выполнение программы в течение заданного числа сканирований

Использование таблицы состояний для контроля и изменения программы

Как показано на рисунке 4–16, для того, чтобы считывать, записывать, принудительно задавать и контролировать переменные во время выполнения программы, вы можете использовать таблицу состояний. Используйте команду меню **View → Status Chart [Вид → Таблица состояний]**.

- Значки панели инструментов таблицы состояний изображаются в области панели инструментов STEP 7-Micro/WIN 32. Эти значки панели инструментов (Sort Ascending [Сортировать по возрастанию], Sort Descending [Сортировать по убыванию], Single Read [Однократное чтение], Write All [Записать все], Force [Активизировать принудительное задание], Unforce [Отменить принудительное задание], Unforce All [Отменить принудительное задание для всех] и Read All Forced [Прочитать все принудительно заданное]) разблокируются, когда вы выбираете таблицу состояний.
- Вы можете создавать несколько диаграмм состояний.
- Для выбора формата ячейки, выделите ячейку, затем нажмите правую кнопку мыши, чтобы разблокировать раскрывающийся список (рис. 4–16).

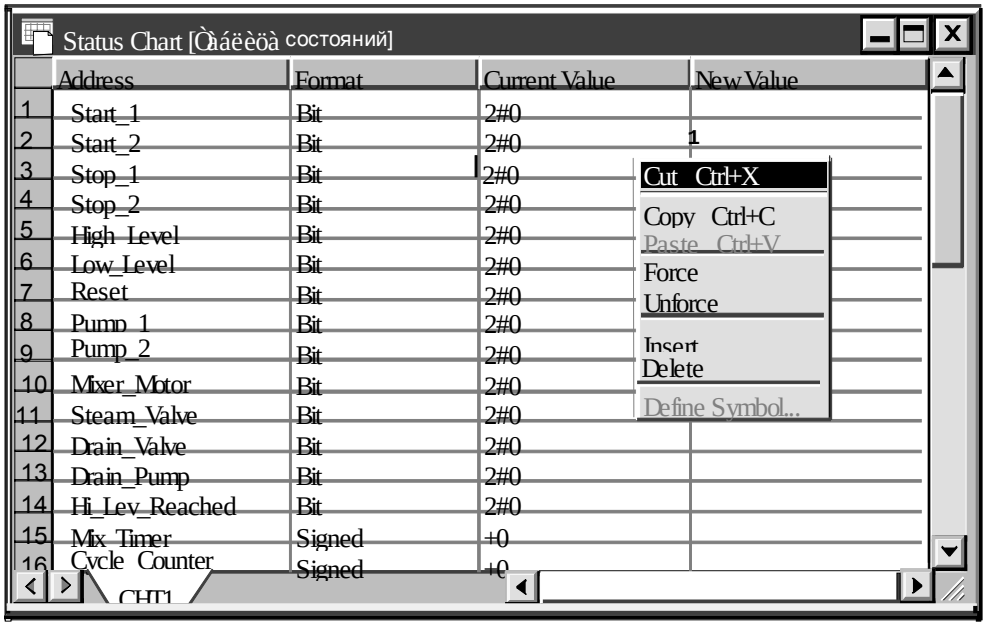


Рис. 4–16. Контроль и изменение переменных через таблицу состояний

Пояснения к рисунку: Address – адрес; Format – формат; Current Value – текущее значение; New Value – новое значение; Cut – вырезать; Copy – копировать; Paste – вставить; Force – принудительно присвоить; Unforce – отменить принудительное присвоение; Insert – вставить; Delete – удалить; Define Symbol – определить символ; High_Level – высокий_уровень; Low_Level – низкий_уровень; Reset – сброс; Pump – насос; Mixer_Motor – двигатель_смесителя; Steam_Valve – паровой_клапан; Drain_Valve – сливной_клапан; Drain_Pumpe – сливной_насос; Hi_Lev_Reached – высокий_уровень_достигнут; Mix_Timer – таймер_миксера; Cycle_Counter – счетчик_циклов.

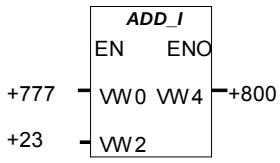
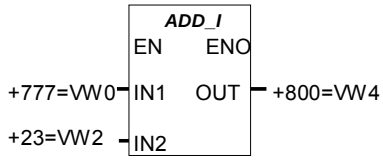
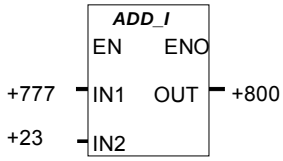
Отображение состояния программы в контактном плане

Вы можете контролировать состояние программы в контактном плане, используя STEP 7-Micro/WIN 32. STEP 7-Micro/WIN 32 должен отображать контактный план. Состояние контактной схемы отображает значения всех операндов команд. См. рис. 4–17. Все состояние основывается на значениях элементов, которые считываются в конце цикла сканирования CPU. STEP 7-Micro/WIN 32 на протяжении нескольких циклов сканирования ПЛК получает значения для отображения состояния, а затем обновляет экранное изображение состояния контактной схемы. Поэтому изображение состояния контактной схемы не отражает текущего состояния каждого элемента контактной схемы во время выполнения.

Для конфигурирования экрана состояния вы можете использовать инструментарий для настройки опций Options Tool. Выберите **Tools → Options [Инструменты → Опции]**, а затем выберите вкладку LAD Status [Состояние LAD]. Таблица 4–13 показывает опции изображения состояния LAD.

Для открытия окна состояния LAD выберите значок состояния на панели инструментов (рис. 4–17).

Таблица 4–13. Выбор опций изображения состояния LAD

Опция изображения	Изображение состояния LAD
Показывать адреса внутри команды, а значения вне команды.	
Показывать адреса и значения вне команды.	
Показ только значение состояния.	

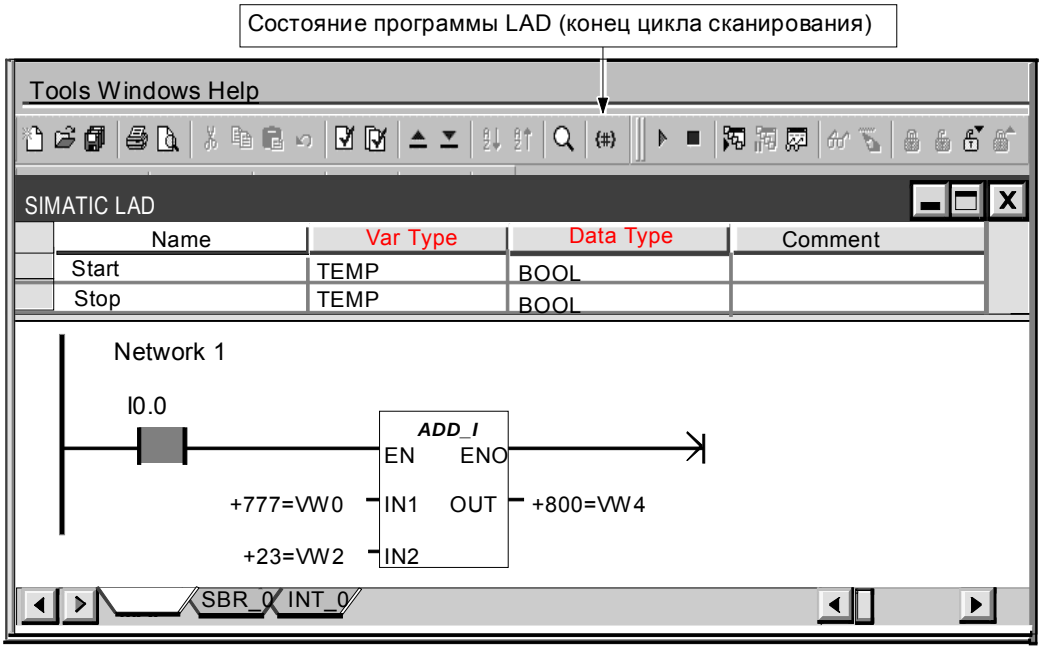


Рис. 4–17 Отображение состояния программы в контактном плане

Пояснения к рисунку: Tools - инструменты; Windows - окна; Help - помощь; Name - имя; Var Type – тип переменной; Data Type – тип данных; Comment - комментарий.

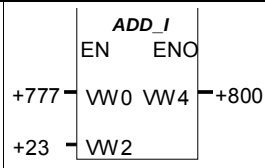
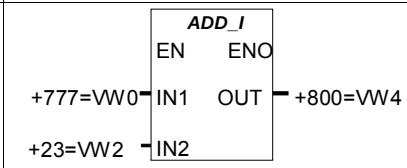
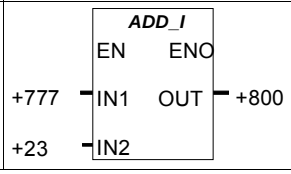
Отображение состояния программы в функциональном плане

Вы можете контролировать состояние программы в FBD, используя STEP 7-Micro/WIN 32. STEP 7-Micro/WIN 32 должен отображать FBD. Состояние FBD отображает значения всех операндов команд. Все состояние основывается на значениях элементов, которые были считаны в конце цикла сканирования ПЛК. STEP 7-Micro/WIN 32 на протяжении нескольких циклов сканирования ПЛК получает значения для отображения состояния, а затем обновляет экранное изображение состояния FBD. Поэтому изображение состояния FBD не отражает текущего состояния каждого элемента FBD во время выполнения.

Для конфигурирования экрана состояния вы можете использовать инструментарий для настройки опций Options Tool. Выберите **Tools → Options [Инструменты → Опции]**, а затем выберите вкладку FBD Status [Состояние FBD]. Таблица 4–14 показывает опции изображения состояния FBD.

Для открытия окна состояния FBD выберите значок состояния на панели инструментов (рис. 4–18).

Таблица 4–14 Выбор опций изображения состояния FBD

Опция изображения	Изображение состояния FBD
Показ адреса внутри команды и значения вне команды.	
Показ адреса и значения вне команды.	
Показ только значения состояния.	

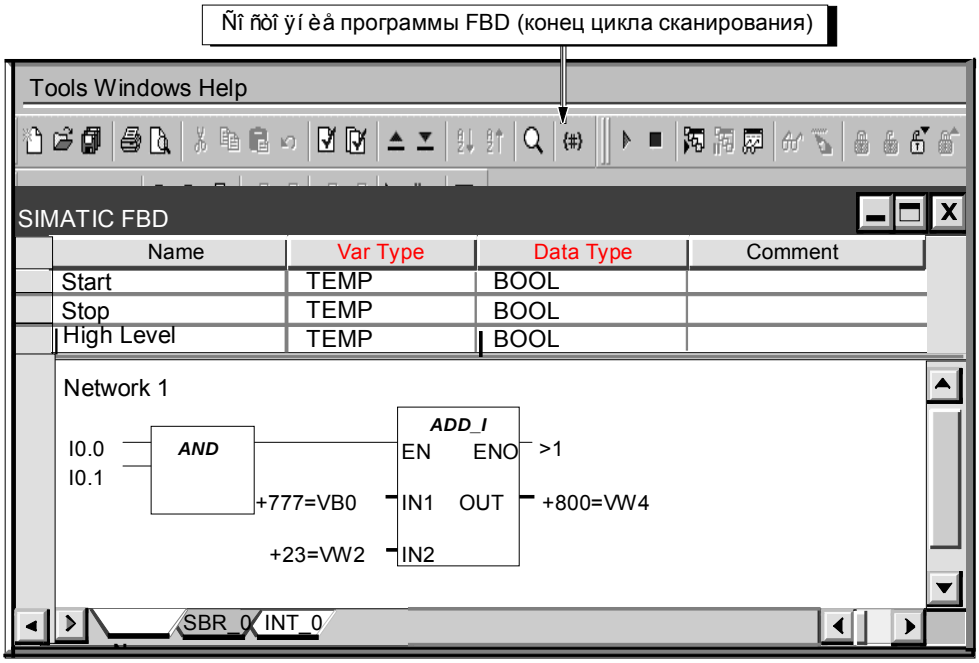


Рис. 4–18 Отображение состояния программы в функциональном плане

Пояснения к рисунку: Tools - инструменты; Windows - окна; Help - помощь; Name - имя; Var Type – тип переменной; Data Type – тип данных; Comment - комментарий.

Отображение состояния программы в форме списка команд

Если вы просматриваете вашу программу, используя редактор STL, то STEP 7-Micro/WIN 32 предоставляет метод контроля состояния выполнения программы, основанный на принципе "команда за командой". Часть вашей программы, для которой включается состояние STL, называется окном состояния STL. Это окно по размеру приблизительно равно вашему экрану STEP 7-Micro/WIN 32. Информация, получаемая от CPU, ограничивается 200 байтами или 25 строками. Появляющиеся на экране строки STL, которые превышают эти пределы, изображаются в окне состояния при помощи знака “-”. Информация о состоянии собирается, начиная с первой команды STL в верхней части окна редактора. Когда вы прокручиваете окно редактора вниз, от CPU собирается новая информация. Типичное окно STL Status [Состояние STL] показано на рисунке 4–19.

Чтобы открыть окно состояния STL, выберите кнопку состояния программы на панели инструментов (рис. 4–19). Доведите до требуемого размера правую границу листа программы STL, чтобы открыть лист состояния STL.

Когда вы включаете состояние STL, на левой стороне окна состояния STL появляется код STL; справа появляется область состояния, содержащая значения операндов. Операнды косвенного адреса отображают и значение по адресу указателя, и адрес указателя. Адрес указателя изображается заключенным в круглые скобки.

Кнопка STL Program Status [Состояние программы STL] непрерывно обновляет значения на экране. Если вы хотите прекратить обновления экрана на следующем обновлении данных, то выделите кнопку Triggered Pause [Включена пауза] (рис. 4–19). Текущие данные будут оставаться на экране, пока вы не снимаете выделение кнопки.

Значения операндов отображаются в столбцах в том порядке, в котором они появляются в каждой команде. Эти значения “захватываются” во время выполнения каждой команды и поэтому отражают текущее состояние выполнения команд.

Цвет значений состояния показывает состояние выполнения команды:

- Черный цвет показывает, что команда выполнена правильно. Безусловные команды, которые не включаются в пределах блока SCR, выполняются всегда, независимо от состояния логического стека. Условные команды для определения выполнения используют логический стек.
- Красный цвет указывает на ошибку выполнения. Обратитесь к описанию конкретной команды в главе 9, где представлены «Ошибки, устанавливающие ENO в 0».
- Серый цвет показывает, что команда не выполнялась, потому что значение вершины стека (s0) = 0 или потому что команда находится внутри блока SCR, который не активизирован.
- Пробел показывает, что команда не выполнялась.

Условия, вызывающие невыполнение команд:

- Вершина логического стека = 0.
- Команда была обойдена вследствие выполнения команд перехода или других конструкций в вашей программе.
- ПЛК не находится в режиме RUN.

Чтобы выделить категории значений, появления которых в окне состояния STL вы желаете, выберите **Tools → Options [Инструменты → Опции]**, а затем выберите вкладку STL Status [Состояние STL]. Вы можете выделить для контроля в окне состояния STL три категории значений:

- операнды (до трех операндов на команду)
- логический стек (до четырех самых последних значений из логического стека)
- биты состояния команды (до двенадцати битов состояния команды)

Чтобы получить информацию о первом цикле сканирования, переключите ПЛК в режим STOP, включите состояние STL, затем выберите **Debug → First Scan [Отладка → Первый цикл сканирования]**.

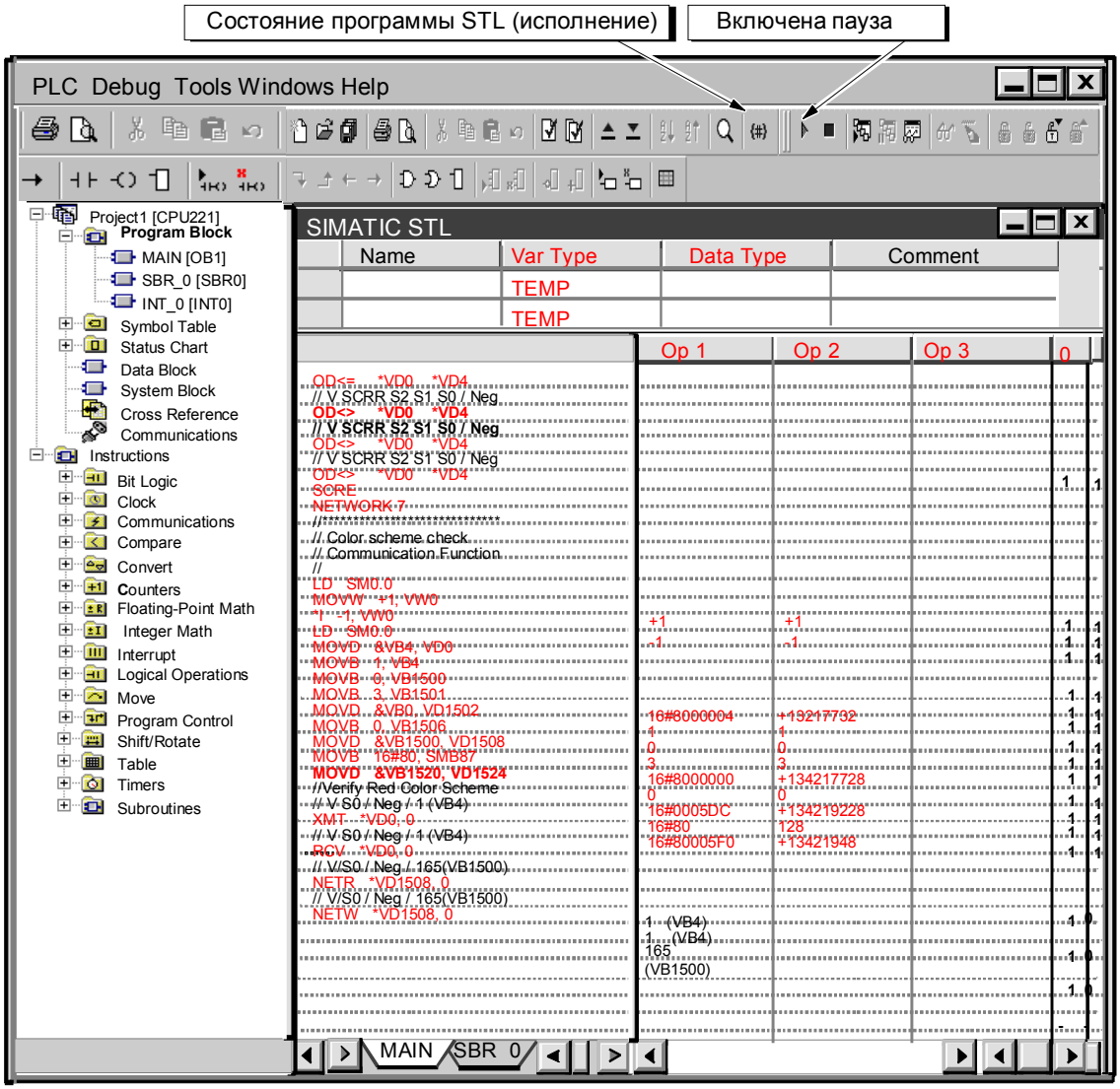


Рис. 4–19. Отображение состояния программы в форме списка команд

Пояснения к рисунку: PLC - ПЛК; Debug – отладка; Tools – инструменты; Windows – окна; Help – помощь; Project – проект; Program Block – программный блок; MAIN – главная программа; SBR - подпрограмма; INT – программа обработки прерывания; Symbol Table – таблица символов; Status Chart – таблица состояний; Data Block – блок данных; System Block – системный блок; Cross Reference – перекрестные ссылки; Communications – обмен данными; Instructions - команды; Bit Logic – битовая логика; Clock – часы; Compare – сравнение; Convert – преобразование; Counters – счетчики; Floating-Point Math – арифметика с плавающей точкой; Integer Math – целая арифметика; Interrupt – прерывание; Logical Operations – логические операции; Move – пересылка; Program Control – управление программой; Shift/Rotate – сдвиг/циклический сдвиг; Table – таблица; Timers – таймеры; Subroutines – подпрограммы.

Принудительное задание значений

CPU S7-200 позволяет принудительно устанавливать значения некоторых или всех входов-входов (биты I и Q). Кроме того, вы можете также принудительно задавать до 16 значений внутренней памяти (V или M) или значений аналоговых входов-выходов (AI или AQ). Значения V-памяти или M-памяти могут принудительно задаваться в форме байтов, слов или двойных слов. Аналоговые значения принудительно задаются как слова только на границах байтов с четными номерами (например, AIW6 или AQW14). Все принудительно задаваемые значения хранятся в постоянной памяти ЭСППЗУ CPU.

Поскольку принудительно установленные данные могут быть изменены во время цикла сканирования (программой, циклом обновления входов-выходов или циклом обработки коммуникационных запросов), CPU повторно применяет принудительно задаваемые значения в различные моменты времени в цикле сканирования. Рис. 4–20 показывает цикл сканирования, подчеркивая, когда CPU обновляет принудительно задаваемые переменные.

Функция принудительного задания подавляет команду непосредственного чтения или команду непосредственной записи. Функция принудительного задания подавляет также настройку выхода на принятие заданного значения при переходе в режим STOP: если CPU переходит в режим STOP, то выход отражает принудительно заданное значение, а не значение настройки.

Как показано на рисунке 4–21, вы можете для принудительного задания значений использовать таблицу состояний (Status Chart). Чтобы принудительно задать новое значение, введите значение в столбец New Value [Новое значение] таблицы состояний, затем нажмите кнопку Force [Принудительное задание] на панели инструментов. Чтобы принудительно задать существующее значение, выделите значение в столбце Current Value [Текущее значение], затем нажмите кнопку Force [принудительное задание].

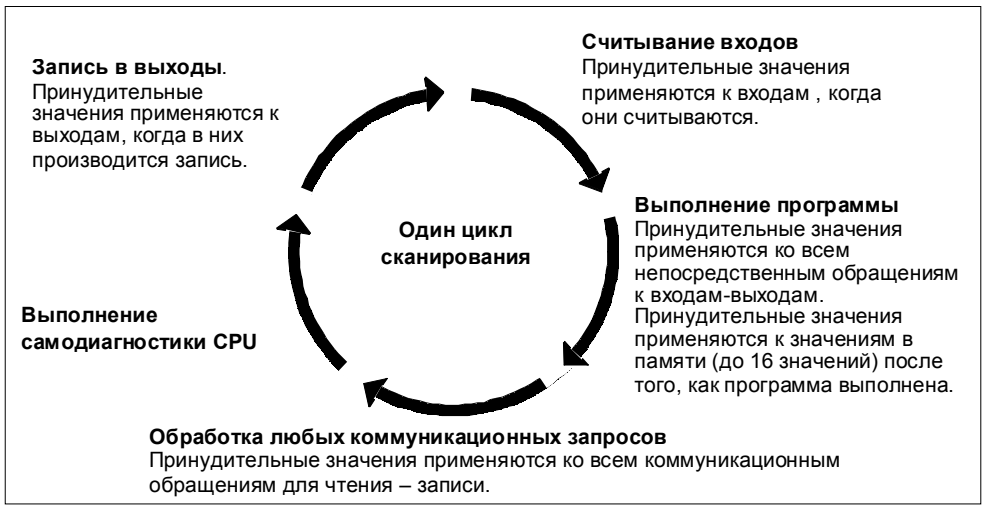


Рис. 4–20. Цикл сканирования CPU S7-200

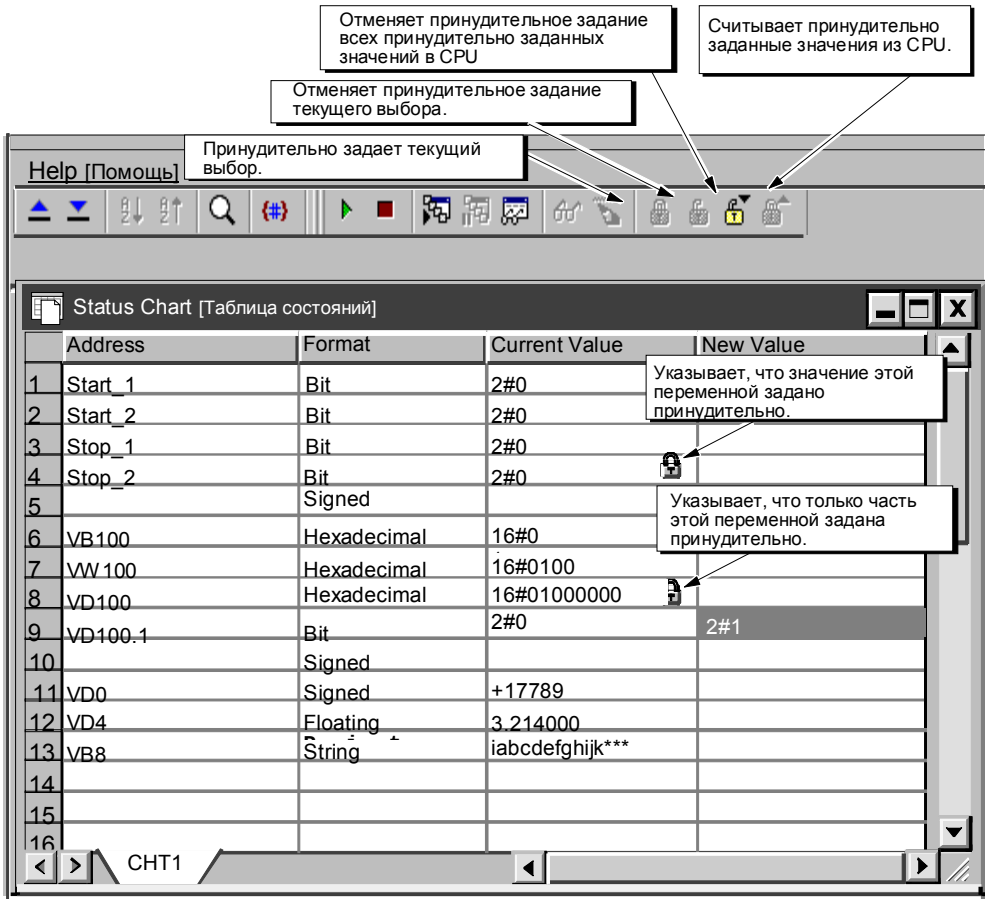


Рис. 4–21. Принудительное задание переменных через таблицу состояний

4.10 Редактирование в режиме RUN

Модели CPU 224 версии 1.10 (и выше) и CPU 226 версии 1.00 (и выше) поддерживают редактирование в режиме RUN. Возможность редактирования в режиме RUN предназначена для того, чтобы позволить вам производить небольшие изменения в пользовательской программе с минимальной помехой для процесса, управляемого программой. Однако реализация этой возможности допускает также крупные изменения программ, которые могут быть разрушительными или даже опасными.

Предупреждение
Когда вы загружаете изменения в CPU в режиме RUN, эти изменения влияют на функционирование процесса. Изменение программы в режиме RUN может немедленно привести к непредвиденному действию системы, способному вызвать серьезную травму или смерть персонала и/или повреждение оборудования. Редактирование в режиме RUN должен выполнять только уполномоченный персонал, понимающий влияние редактирования в режиме RUN на работу системы.

Для выполнения редактирования программы в режиме RUN должны быть выполнены следующие условия:

- Онлайновый CPU должен поддерживать редактирование в режиме RUN.
- Онлайновый CPU должен находиться в режиме RUN.

Для осуществления редактирования в режиме RUN, выполните следующие шаги:

1. Выберите **Debug → Program Edit in RUN [Отладка → Редактирование программы в RUN]**. (Рис. 4–22 показывает меню Debug [Отладка] на правой стороне экрана.)
2. Если проект отличается от программы в CPU, то вы получаете приглашение сохранить его. Редактирование в режиме RUN может выполняться только в программе, находящейся в CPU.

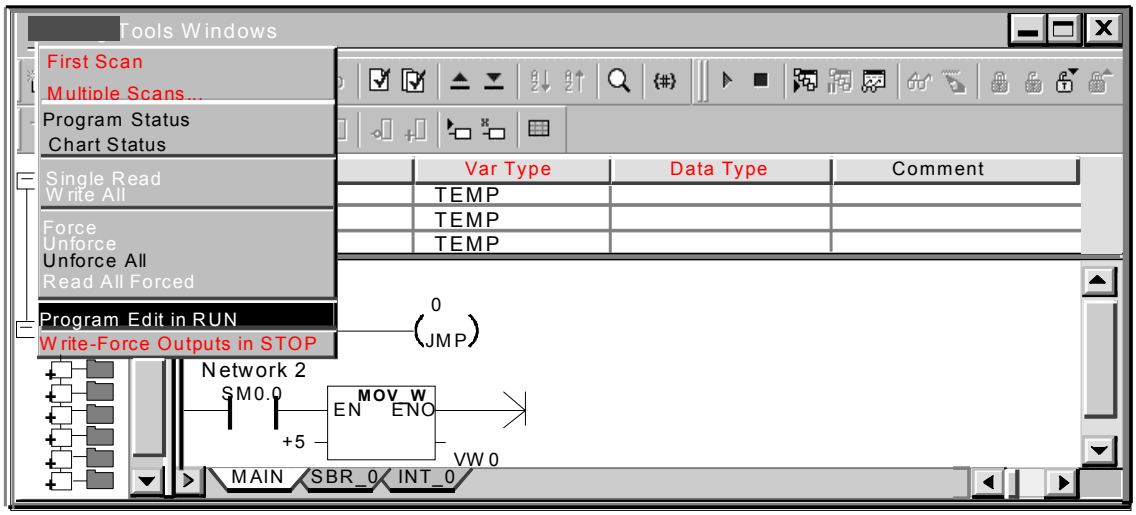


Рис. 4–22. Отображение состояния программы в функциональном плане

Пояснения к рисунку: Debug - отладка; Tools - инструменты; Windows - окна; First Scan – первый цикл сканирования; Multiple Scans – несколько циклов сканирования; Program Status – состояние программы; Chart Status – таблица состояний; Single Read – однократное считывание ; Write All – записать все; Force – принудительное задание; Unforce – отмена принудительного задания; Unforce All – отмена всех принудительных заданий; Read All Forced – чтение всех значений, заданных принудительно; Program Edit in RUN – редактирование программы в RUN; Write-Force Outputs in STOP – принудительная запись выходов в STOP.

3. Появляется предупреждение, показанное на рисунке 4–23. Когда вы выбираете кнопку “Continue [Продолжить]”, загружается программа из подключенного CPU, и приложение находится в состоянии редактирования в режиме RUN. Никакие ограничения на редактирование не накладываются.

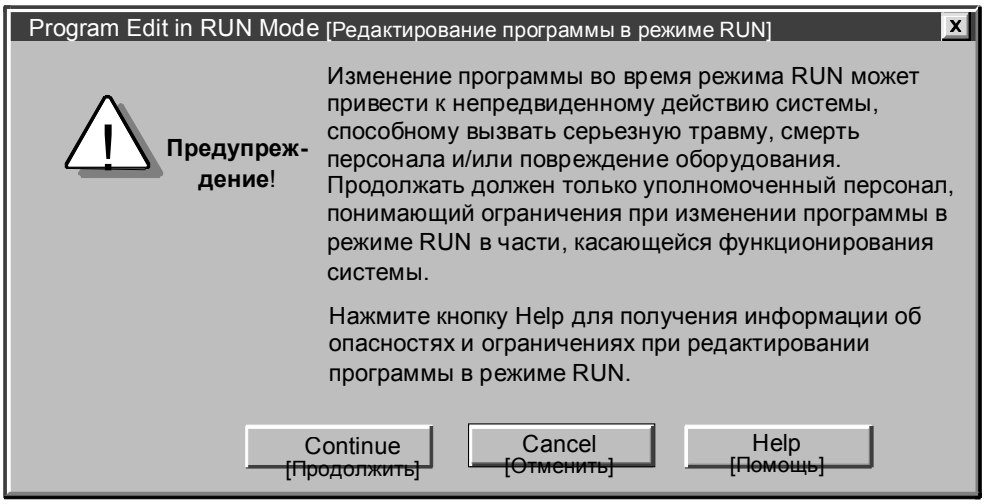


Рис. 4–23. Диалоговое окно редактирования в режиме RUN

Примечание
Команды оценки положительного (EC) и отрицательного (ED) фронта изображаются с операндом. Для просмотра информации о предыдущем состоянии команд оценки фронта выберите значок Cross Reference [Перекрестные ссылки] в разделе View [Вид] вашего экрана. Вкладка Edge Usage [Использование фронта] перечисляет номера команд оценки фронта в вашей программе. Будьте внимательны, чтобы не назначить двойные номера фронтов, когда вы редактируете свою программу.

Прежде чем вы выполните загрузку в режиме RUN

Редактирование в режиме RUN позволяет загружать новые редакции вашей программы, когда CPU находится в режиме RUN. Чтобы решить, загружать ли вашу программу в CPU в режиме RUN или в режиме STOP, рассмотрите следующие факты:

- Если при редактировании в режиме RUN удаляется управляющая логика для выхода, то выход сохраняет свое последнее состояние до следующего выключения и последующего включения питания или перехода в режим STOP.
- Если при редактировании в режиме RUN удаляются выполнявшиеся функции HSC или PTO/PWM, то эти функции продолжают работать до следующего выключения и последующего включения питания или перехода в режим STOP.
- Если при редактировании в режиме RUN удаляются команды ATCH, но сама программа обработки прерываний не удаляется, то прерывание продолжает выполняться до выключения и последующего включения питания или перехода в режим STOP. Аналогично, если удаляются команды DTCH, то прерывания не отключаются до следующего выключения и последующего включения питания или перехода в режим STOP.
- Если при редактировании в режиме RUN добавляются команды ATCH, условием выполнения которых является установка бита первого цикла сканирования, то эти события не активизируются до следующего выключения и последующего включения питания или перехода из режима STOP в RUN.
- Если при редактировании в режиме RUN удаляется команда ENI, то прерывания продолжают работать до следующего выключения и последующего включения питания или перехода из режима RUN в STOP.
- Если при редактировании в режиме RUN изменяется адрес таблицы блока приема, и блок приема активен, когда выполняется переключение со старой программы на новую, то принимаемые данные записываются по старому адресу таблицы. Команды NETR и NETW будут функционировать аналогично.
- Логический оператор, действие которого обусловлено установкой бита первого цикла сканирования, не выполняется до тех пор, пока не введен режим RUN как результат выключения и последующего включения питания или перехода из режима STOP в RUN, потому что редактирование в режиме RUN не влияет на бит первого цикла сканирования.

Загрузка в режиме RUN

Чтобы загрузить программу в режиме RUN, выберите кнопку загрузки на панели инструментов или выберите в строке меню **File → Download**

[Файл → Загрузить]. Если программа компилируется успешно, то она загружается в CPU. Ход процесса в CPU отображается на индикаторе выполнения. Заметьте, что вы можете загружать в CPU только блок программы. Чтобы загрузка была разрешена, должны быть выполнены следующие условия:

- CPU должен поддержать редактирование в режиме RUN.
- Операция компиляции должна быть завершена без ошибок.
- Обмен данными между компьютером, на котором исполняется STEP 7 Micro/WIN 32, и CPU должен происходить без ошибок.

Выход из редактирования в режиме RUN

Для выхода из редактирования в режиме RUN выберите **Debug → Program Edit in RUN [Отладка → Редактирование программы в RUN]** и щелкните по метке, чтобы удалить ее. Если у вас имеются изменения, которые не были сохранены, то вы можете выбрать продолжение редактирования, либо загрузку изменений и выход из редактирования в режиме RUN, либо выход без загрузки.

4.11 Фоновое время

Вы можете задать долю цикла сканирования в процентах, которая будет выделяться обработке коммуникационных запросов, связанных с компиляцией при редактировании в режиме RUN или сбором информации о состоянии STL. Когда вы увеличиваете процент времени, выделяемого обработке коммуникационных запросов, вы увеличиваете время сканирования, что замедляет выполнение процесс управления.

Чтобы задать квант времени цикла сканирования, разрешенный для фонового обмена данными, выберите вкладку Background Time [Фоновое время] (рис. 4–24). Отредактируйте характеристики коммуникационного фонового времени и затем загрузите изменения в CPU.

Процент времени сканирования, выделяемый обработке коммуникационных запросов, по умолчанию устанавливается равным 10 %. Такая настройка была выбрана для того, чтобы обеспечить разумный компромисс для обработки операций компиляции/вывода состояния при минимизации влияния на ваш процесс управления. Вы можете корректировать это значение с шагом 5 % до максимального значения 50 %.

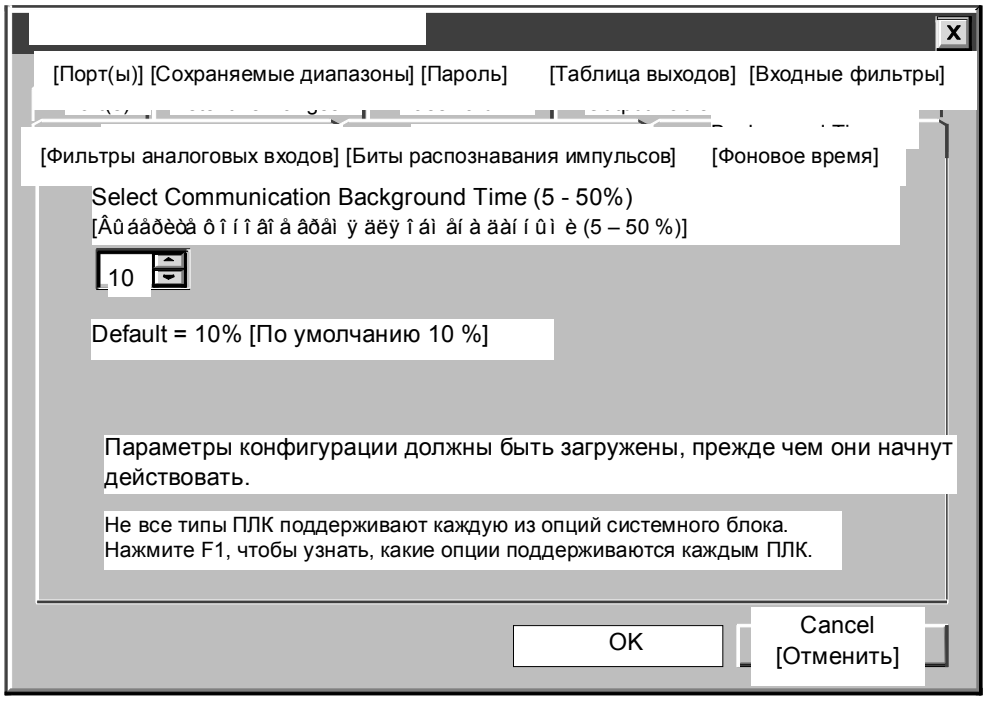


Рис. 4–24. Экран настройки фонового времени

4.12 Обработка ошибок CPU S7-200

CPU S7-200 разделяет ошибки на фатальные и нефатальные. Для просмотра кодов, порождаемых ошибкой, вы можете использовать STEP 7-Micro/WIN 32. Чтобы просмотреть эти ошибки, выберите в строке меню пункт **PLC→ Information [ПЛК → Информация]**. На рис. 4–25 показано диалоговое окно, содержащее код и описание ошибки. За полным списком кодов ошибки обратитесь к Приложению B.

На рисунке 4–25 поле Last Fatal [Последняя фатальная ошибка] показывает код предыдущей фатальной ошибки, сформированный CPU. Это значение сохраняется при выключениях и включениях питания, если сохраняется ОЗУ. Эта ячейка очищается всякий раз, когда очищается вся память CPU, или когда ОЗУ не сохраняется после длительного перерыва в подаче питания.

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

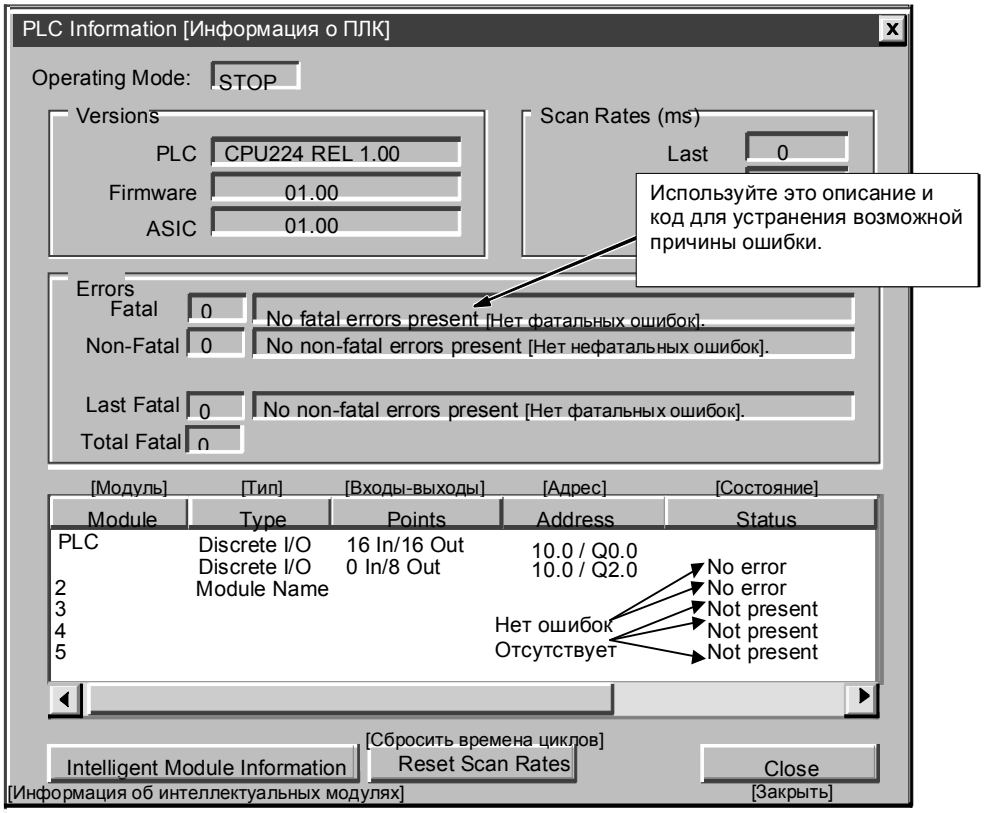


Рис. 4–25. Диалоговое окно информации CPU: Вкладка Error Status [Состояние ошибки]

Пояснения к рисунку: Operating Mode – режим работы; Versions – версии; PLC – ПЛК; Firmware – встроенные программы; ASIC – специализированная интегральная схема; Scan Rates – времена циклов; Last - последний; Errors - ошибки; Fatal - фатальная; Non-Fatal - нефатальная; Last Fatal – последняя фатальная; Total Fatal – всего фатальных; Discrete I/O – дискретный ввод-вывод; Module Name – имя модуля; In - вход; Out - выход.

Реакция на фатальные ошибки

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

Как только вы сделали изменения для устранения фатальной ошибки, вы должны перезапустить CPU. Вы можете перезапустить CPU, используя один из следующих методов:

- Выключение и затем включение питания.
- Перевод переключателя режимов работы из RUN или TERM в STOP.
- Использование STEP 7-Micro/WIN для перезапуска CPU. STEP 7-Micro/WIN 32 осуществляет **PLC → Power–Up Reset [ПЛК → Сброс включения питания]** из строки главного меню. Это заставляет ПЛК перезапуститься и сбросить все фатальные ошибки.

Перезапуск CPU 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 CPU снова устанавливает светодиод, показывая, что ошибка по-прежнему существует. В противном случае CPU начинает нормальную работу.

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

Реакция на нефатальные ошибки

Нефатальные ошибки могут ухудшать некоторые характеристики CPU, но они не делают CPU неспособным к выполнению вашей программы или обновлению входов-выходов. Как показано на рисунке 4–25, вы можете использовать STEP 7-Micro/WIN 32, для просмотра кодов, порождаемых нефатальной ошибкой. Имеются три основные категории нефатальных ошибок:

- Ошибки этапа выполнения. Все нефатальные ошибки, обнаруживаемые в режиме RUN отображаются в битах специальной памяти (SM). Ваша программа может контролировать и анализировать эти биты. Для получения дополнительной информации о битах SM, используемых для сообщения о нефатальных ошибках этапа выполнения, обратитесь к Приложению C.

  При запуске CPU считывает конфигурацию входов-выходов и сохраняет эту информацию в системной памяти данных и в памяти SM. Во время нормальной работы состояние входов-выходов периодически обновляется и сохраняется в памяти SM. Если CPU обнаруживает отличие в конфигурации входов-выходов, то он устанавливает бит «конфигурация изменена» в байте ошибок модулей; пока этот бит не сброшен, модуль ввода-вывода не обновляется. Чтобы CPU сбросил этот бит, модуль ввода-вывода должен снова соответствовать конфигурации входов-выходов, хранимой в системной памяти данных.
- Ошибки компиляции программы. CPU компилирует программу, когда он ее загружает. Если CPU обнаруживает, что программа нарушает правило компиляции, то загрузка прерывается и генерируется код ошибки. (Программа, которая уже была загружена в CPU, по-прежнему будет существовать в ЭСППЗУ и не потеряется.) После исправления своей программы вы можете загрузить ее снова.
- Ошибки выполнения программы. Ваша программа может создавать состояния ошибки во время своего выполнения. 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 считается ошибкой программирования, проявляющейся при выполнении программы. Чтобы определить, какого типа ошибка произошла, используйте диалоговое окно, показанное на рис. 4–25 на странице 4–44.

Когда CPU 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 CPU к переключению в режим STOP, когда обнаруживается нефатальная ошибка. Рис. 4–26 показывает сегмент программы, который контролирует бит SM. Эта команда переключает CPU в режим STOP всякий раз, когда обнаруживается ошибка ввода-вывода.

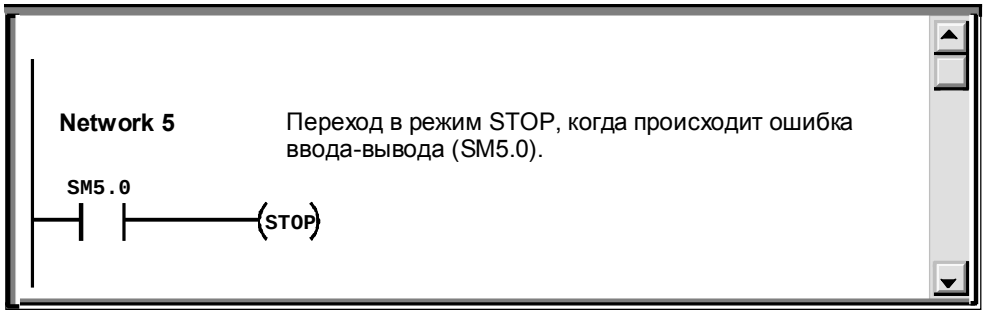


Рис. 4–26. Проектирование программа для обнаружения нефатальных ошибок