Основы ПЛК

4

Основной функцией S7–200 является контроль полевых входов и, на основе логики управления, включение и выключение полевых выходных устройств. В этой главе объясняются основы выполнения программы, различные виды используемой памяти и способы сохранения.

В этой главе

Выполнение логики управления с помощью S7–200

S7–200 обрабатывает логику управления в вашей программе циклически, считывая и записывая данные.

S7–200 ставит вашу программу в соответствие физическим входам и выходам

Основной принцип действия S7–200 очень прост:

- S7–200 считывает состояние входов.
- Программа, хранящаяся в S7–200, использует эти входы для анализа логики управления. Во время обработки программы S7–200 обновляет данные.
- S7–200 записывает данные на выходы.

На рис. 4–1 показана связь между простой коммутационной схемой и S7–200. В этом примере состояние выключателя для запуска двигателя логически связано с состояниями других входов. Оценки этих состояний определяют затем сигнальное состояние выхода для исполнительного устройства, которое запускает двигатель.

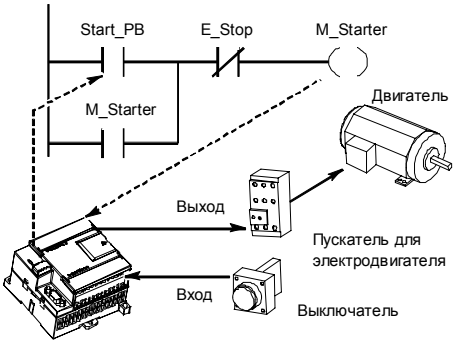


Рис. 4–1. Управление входами и выходами

S7–200 выполняет все задачи в цикле

S7–200 выполняет последовательность задач неоднократно. Эта регулярная обработка задач называется циклом. Как показано на рис. 4–2, S7–200 выполняет в цикле большинство или все из следующих задач:

- Чтение входов: S7–200 копирует состояние физических входов в регистр входов образа процесса.
- Выполнение логики управления в программе: S7–200 выполняет команды программы и сохраняет значения в различных областях памяти.
- Обработка запросов на обмен данными: S7–200 выполняет все задачи, необходимые для обмена данными.
- Самодиагностика CPU: S7–200 проверяет, чтобы встроенное программное обеспечение, программная память и все модули расширения работали надлежащим образом.
- Запись в выходы: Значения, хранящиеся в регистре выходов образа процесса, записываются в физические выходы.

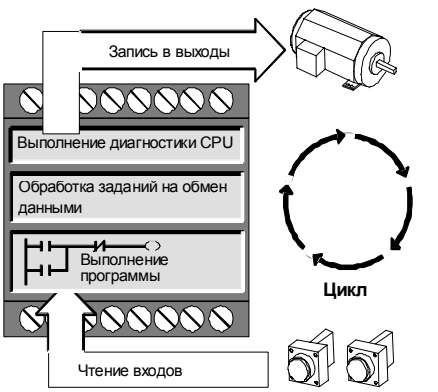


Рис. 4–2. Цикл S7–200

Выполнение цикла зависит от того, находится ли S7–200 в состоянии STOP или в состоянии RUN. В состоянии RUN ваша программа выполняется; в состоянии STOP ваша программа не выполняется.

Чтение входов

Цифровые входы: В начале цикла текущие значения цифровых входов считываются, а затем записываются в регистр входов образа процесса.

Аналоговые входы: S7–200 не обновляет аналоговые входы автоматически как часть цикла, если вы не активизировали фильтрацию аналоговых входов. Аналоговый фильтр обеспечивает стабильность сигналов. Вы можете активизировать аналоговый фильтр для каждого входа.

Если фильтр для аналогового входа активизирован, то S7–200 обновляет этот аналоговый вход один раз за цикл, выполняет функцию фильтрации и сохраняет отфильтрованное значение внутри. Это отфильтрованное значение затем предоставляется в распоряжение всякий раз, когда ваша программа обращается к этому аналоговому входу.

Если фильтр аналогового входа выключен, то S7–200 считывает значение этого аналогового входа из физического модуля всякий раз, когда ваша программа обращается к аналоговому входу.

Совет

Фильтр аналогового входа обеспечивает стабильность аналоговых значений. Фильтр аналогового входа следует активизировать в приложениях, в которых входной сигнал медленно меняется с течением времени. Если речь идет о быстро меняющемся сигнале, то аналоговый фильтр активизировать не следует.

Не применяйте аналоговый фильтр у модулей, которые передают цифровые данные или сигналы тревоги в аналоговых словах. Всегда выключайте аналоговый фильтр для ведущих модулей с RTD, термопарами и AS–интерфейсом.

Обработка программы

На этом участке цикла S7–200 обрабатывает программу с первой команды до последней. Вы можете непосредственно управлять входами и выходами и получать, таким образом, доступ к ним во время исполнения основной программы или программы обработки прерываний.

Если вы используете в своей программе прерывания, то программы обработки прерываний, которые ставятся в соответствие прерывающим событиям, хранятся как часть основной программы. Однако программы обработки прерываний исполняются не как составная часть нормального цикла, а только тогда, когда происходит прерывающее событие (оно возможно в любом месте цикла).

Обработка запросов на обмен данными

На участке цикла, выделенном для обработки коммуникаций, S7–200 обрабатывает все сообщения, полученные из коммуникационного порта или от интеллектуальных модулей ввода/вывода.

Самодиагностика CPU

На этом участке цикла S7–200 проверяет надлежащую работу CPU, области памяти и состояние модулей расширения.

Запись в цифровые выходы

В конце каждого цикла S7–200 записывает значения, хранящиеся в регистре выходов образа процесса, в цифровые выходы. (Аналоговые выходы обновляются немедленно, независимо от цикла.)

Доступ к данным S7–200

S7–200 хранит информацию в различных местах памяти, которые имеют однозначные адреса. Вы можете явно указать адрес в памяти, к которому вы хотите обратиться. Благодаря этому ваша программа имеет прямой доступ к информации. Таблица 4–1 показывает диапазон целых значений, которые могут быть представлены с помощью данных различной длины.

Таблица 4–1. Десятичные и шестнадцатеричные диапазоны для данных различной длины

Представление	Байт (B)	Слово (W)	Двойное слово (D)
Целое без знака	от 0 до 255 от 0 до FF	от 0 до 65 535 от 0 до FFFF	от 0 до 4 294 967 295 от 0 до FFFF FFFF
Целое со знаком	от -128 до +127 от 80 до 7F	от –32 768 до +32 767 от 8000 до 7FFF	от –2 147 483 648 до +2 147 483 647 от 8000 0000 до 7FFF FFFF
Вещественное IEEE 32–битовое с плавающей точкой	*Неприменимо*	*Неприменимо*	от +1.175495E-38 до +3.402823E+38 (положительное) от –1.175495E-38 до –3.402823E+38 (отрицательное)

Для обращения к биту в некоторой области памяти вы должны указать адрес бита. Этот адрес состоит из идентификатора области памяти, адреса байта и номера бита. На рис. 4–3 показан пример обращения к биту (адресация в формате «байт.бит»). В этом примере за областью памяти и адресом байта (I = input [вход], 3 = байт 3) следует точка («.»), чтобы отделить адрес бита (бит 4).

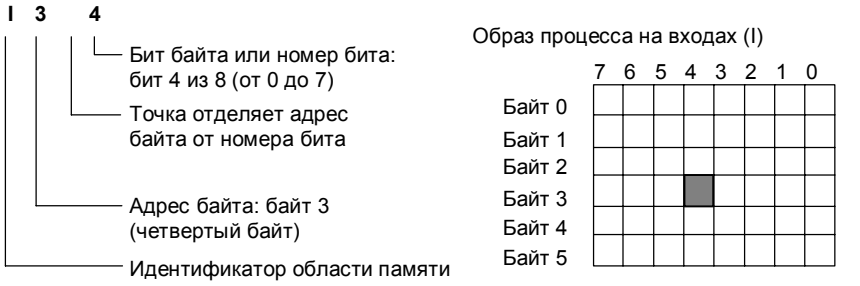


Рис. 4–3. Адресация байт.бит

Когда вы применяете формат байт.бит, вы можете обратиться к данным в большинстве областей памяти (V, I, Q, M, S, L и SM) как к байтам, словам или двойным словам. Если вы хотите обратиться к байту, слову или двойному слову данных в памяти, то вы должны указать эти адреса подобно адресу бита. Вы указываете идентификатор области, обозначение длины данных и начальный адрес байта, слова или двойного слова, как показано на рис. 4–4.

К данным в других областях памяти (напр., T, C, HC и аккумуляторы) вы обращаетесь, указывая в качестве адреса идентификатор области и номер элемента.

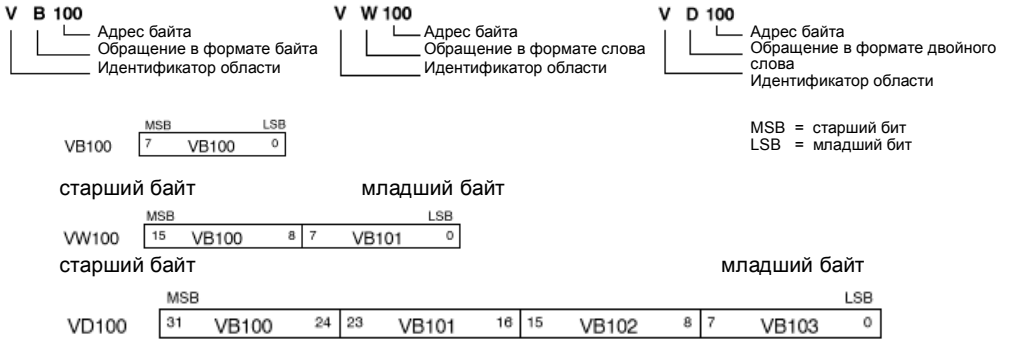


Рис. 4–4. Обращение к одному и тому же адресу в формате байта, слова и двойного слова

Обращение к данным в областях памяти

Регистр входов образа процесса: I

В начале каждого цикла S7–200 опрашивает физические входы и записывает полученные значения во регистр входов образа процесса. К образу процесса можно обратиться в формате бита, байта, слова и двойного слова:

Бит:	I*[адрес байта].[адрес бита]*	I0.1
Байт, слово или двойное слово:	I*[длина][начальный адрес байта]*	IB4

Регистр выходов образа процесса: Q

В конце цикла S7–200 копирует значения, хранящиеся в регистре выходов образа процесса, в физические выходы. К образу процесса можно обратиться в формате бита, байта, слова и двойного слова:

Бит:	Q*[адрес байта].[адрес бита]*	Q1.1
Байт, слово или двойное слово:	Q*[длина][начальный адрес байта]*	QB5

Область памяти переменных: V

Память переменных можно использовать для хранения промежуточных результатов операций, выполняемых в вашей программе. В памяти переменных вы можете хранить также другие данные, имеющие отношение к процессу или к решению вашей задачи автоматизации. К памяти переменных можно обратиться в формате бита, байта, слова и двойного слова:

Бит:	V*[адрес байта].[адрес бита]*	V10.2
Байт, слово или двойное слово:	V*[длина][начальный адрес байта]*	VW100

Область битовой памяти: M

Биты памяти (меркеры) можно использовать как управляющие реле для хранения промежуточных результатов операций или другой управляющей информации. К битам памяти можно обратиться в формате бита, байта, слова и двойного слова:

Бит:	M*[адрес байта].[адрес бита]*	M26.7
Байт, слово или двойное слово:	M*[длина][начальный адрес байта]*	MD20

Таймеры: T

S7–200 имеет в своем распоряжении таймеры, которые отсчитывают приращения времени с разрешениями (шагами базы времени) 1 мс, 10 мс или 100 мс. С таймером связаны две переменные:

- Текущее значение: это 16–битовое целое со знаком хранит количество времени, отсчитанное таймером (значение таймера).
- Бит таймера: этот бит устанавливается или сбрасывается, когда текущее значение становится равным предустановленному значению. Предустановленное значение вводится как часть таймерной команды.

Вы обращаетесь к обоим этим элементам данных через адрес таймера (T + номер таймера). Происходит ли обращение к биту таймера или к текущему значению, зависит от используемой команды: команды с операндами в битовом формате обращаются к биту таймера, тогда как команды с операндами в формате слова обращаются к текущему значению. Как показано на рис. 4–5, команда "Нормально открытый контакт" обращается к биту таймера, а команда "Передать слово" обращается к текущему значению таймера.

Формат: T*[номер таймера]* T24

Рис. 4–5. Обращение к биту или к текущему значению таймера

Счетчики: C

S7–200 имеет в своем распоряжении три вида счетчиков, которые подсчитывают нарастающие фронты на счетных входах счетчика: один вид счетчиков ведет прямой счет, другой считает только в обратном направлении, а третий вид считает в обоих направлениях. Со счетчиком связаны две переменные:

- Текущее значение: это 16–битовое целое со знаком хранит счетное значение, накопленное счетчиком.
- Бит счетчика: этот бит устанавливается или сбрасывается, когда текущее значение становится равным предустановленному значению. Предустановленное значение вводится как часть команды счетчика.

Вы обращаетесь к обоим этим элементам данных через адрес счетчика (C + номер счетчика). Происходит ли обращение к биту счетчика или к текущему значению, зависит от используемой команды: команды с операндами в битовом формате обращаются к биту счетчика, тогда как команды с операндами в формате слова обращаются к текущему значению. Как показано на рис. 4–6, команда "Нормально открытый контакт" обращается к биту счетчика, а команда "Передать слово" обращается к текущему значению счетчика.

Формат: C*[номер счетчика]* C24

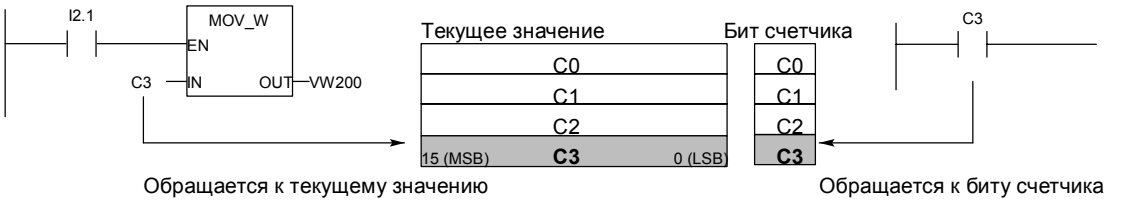


Рис. 4–6. Обращение к биту или к текущему значению счетчика

Скоростные счетчики: HC

Скоростные счетчики подсчитывают быстрые события независимо от цикла CPU. Скоростные счетчики имеют в своем распоряжении 32–битовое целое счетное значение (текущее значение). Для обращения к счетному значению скоростного счетчика введите его адрес, указав область памяти (HC) и номер счетчика (напр., HC0). Текущее значение скоростного счетчика защищено от записи и может быть адресовано только в формате двойного слова (32 бита).

Формат: HC*[номер скоростного счетчика]* HC1

Аккумуляторы: AC

Аккумуляторы – это элементы чтения/записи, которые могут использоваться как память. Например, вы можете использовать аккумуляторы для передачи параметров в подпрограммы и из них или для хранения промежуточных результатов расчетов. S7–200 имеет в своем распоряжении четыре 32–битовых аккумулятора (AC0, AC1, AC2 и AC3). К данным в аккумуляторах можно обратиться в формате бита, слова или двойного слова.

Длина данных, к которым производится обращение, зависит от команды, которая используется для обращения к аккумулятору. Как показано на рис. 4–7, при обращении к аккумулятору в формате бита или слова используются младшие 8 или 16 битов значения, хранящегося в аккумуляторе. При обращении к аккумулятору в формате двойного слова используются все 32 бита.

Информацию об использовании аккумуляторов в программах обработки прерываний вы найдете в главе 6.

Формат: AC*[номер аккумулятора]* AC0

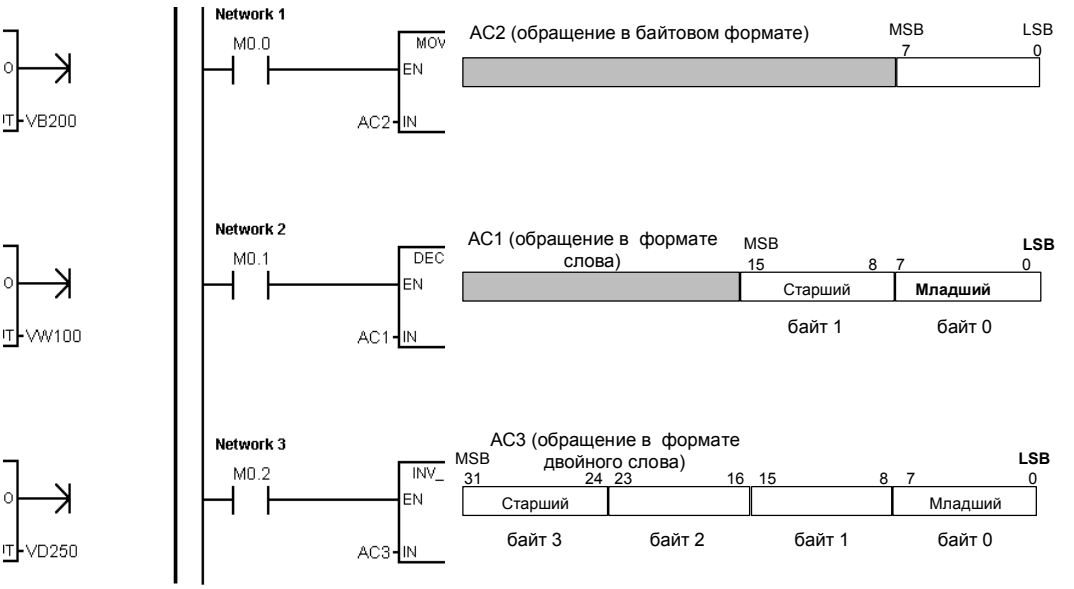


Рис. 4–7. Обращение к аккумуляторам

Специальные биты памяти: SM

Специальные биты памяти (SM) предоставляют средство для обмена данными между CPU и вашей программой. Вы можете использовать эти биты для выбора и управления некоторыми специальными функциями CPU S7–200, например: бит, который устанавливается только в первом цикле; бит, который устанавливается и сбрасывается с фиксированной частотой, или бит, который указывает на состояние арифметической или иной команды. (Подробную информацию о специальных битах памяти вы найдете в Приложении D.) К SM-битам можно обращаться в формате бита, слова или двойного слова:

Бит:	SM*[адрес байта].[адрес бита]*	SM0.1
Байт, слово или двойное слово:	SM*[длина][начальный адрес байта]*	SMB86

Память локальных данных: L

S7–200 имеет в своем распоряжении 64 байта локальной памяти, из которых 60 могут быть использованы в качестве промежуточной памяти или для передачи формальных параметров в подпрограммы.

Совет
При программировании в LAD или FBD последние четыре байта зарезервированы для STEP 7-Micro/WIN. Если вы программируете на STL, то все 64 байта локальной памяти имеются в вашем распоряжении, но мы рекомендуем вам и в этом случае не использовать эти последние четыре байта.

Память локальных данных похожа на память переменных с одним существенным отличием. Память переменных доступна глобально, тогда как память локальных данных доступна локально. Глобальная доступность означает, что к адресу в этой области памяти можно обратиться из любой организационной единицы программы (из основной программы, подпрограммы или подпрограмм обработки прерываний). Локальная доступность означает, что эта область памяти ставится в соответствие определенной организационной единице программы. S7–200 выделяет 64 байта локальной памяти для главной программы, 64 байта для каждого уровня вложенности подпрограмм и 64 байта для программ обработки прерываний.

К области локальных данных, поставленной в соответствие основной программе, не имеют доступа подпрограмм и программы обработки прерываний. Подпрограмма не может обращаться к области локальных данных основной программы, программы обработки прерываний или другой подпрограммы. Аналогично, программа обработки прерываний не имеет доступа к области локальных данных основной программы или подпрограммы.

S7–200 выделяет область локальных данных по мере необходимости. Это значит, что при выполнении основной программы области локальных данных для подпрограмм и программ обработки прерываний не существуют. Если возникает прерывание или вызывается подпрограмма, то по потребности выделяется локальная память. Вновь выделенная локальная память может снова использовать те же адреса, которые использовались другой подпрограммой или программой обработки прерываний.

S7–200 не инициализирует область локальных данных к моменту ее назначения, поэтому она может содержать любые значения. Если при вызове подпрограммы передаются формальные параметры, то S7–200 сохраняет значения передаваемых параметров в соответствующих адресах области локальных данных, выделенной этой подпрограмме. Адреса в области локальных данных, которые не получили значений при передаче формальных параметров, не инициализируются и при выделении могут содержать произвольные значения.

Бит:	L*[адрес байта].[адрес бита]*	L0.0
Байт, слово или двойное слово:	L*[длина] [начальный адрес байта]*	LB33

Аналоговые входы: AI

S7–200 преобразует аналоговые величины (например, температуру или напряжение) в цифровые величины, имеющие длину слова (16 битов). Вы обращаетесь к этим значениям через идентификатор области (AI), длину данных (W) и начальный адрес байта. Так как в случае аналоговых входов речь идет о словах, которые всегда начинаются на байтах с четными номерами (например, 0, 2, 4 и т.д.), то вы обращаетесь к этим значениям с помощью адресов четных байтов (например, AIW0, AIW2, AIW4). Аналоговые входы можно только считывать.

Формат:	AIW*[начальный адрес байта]*	AIW4

Аналоговые выходы: AQ

S7–200 преобразует цифровые величины, имеющие длину слова (16 битов), в ток или напряжение пропорционально цифровой величине. Вы обращаетесь к этим значениям через идентификатор области (AQ), длину данных (W) и начальный адрес байта. Так как в случае аналоговых выходов речь идет о словах, которые всегда начинаются на байтах с четными номерами (например, 0, 2, 4 и т.д.), то вы записываете эти значения с адресами четных байтов (например, AQW0, AQW2, AQW4). Аналоговые выходы можно только записывать.

Формат:	AQW*[начальный адрес байта]*	AQW4

Реле управления очередностью (SCR): S

SCR или S-биты разделяют функционирование установки на отдельные шаги или эквивалентные части программы. С помощью реле управления очередностью программа управления представляется в виде структуры, состоящей из логических сегментов. К S-битам можно обращаться в формате бита, слова или двойного слова.

Бит:	S*[адрес байта].[адрес бита]*	S3.1
Байт, слово или двойное слово:	S*[длина][начальный адрес байта]*	SB4

Формат вещественных чисел

Вещественные числа (или числа с плавающей точкой) представляются как 32–битовые числа однократной точности, формат которых описан в стандарте ANSI/IEEE 754-1985. См. рис. 4–8. Обращение к вещественным числам производится в формате двойного слова.

У S7–200 числа с плавающей точкой имеют точность до 6 десятичных разрядов. Поэтому при вводе константы с плавающей точкой можно указывать до 6 десятичных разрядов.

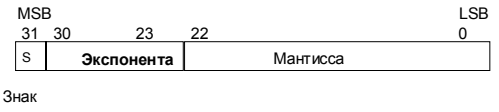


Рис. 4–8. Формат вещественного числа

Точность при вычислениях с вещественными числами

Расчеты, включающие в себя длинные последовательности значений, содержащие очень большие и очень малые числа, могут привести к неточным результатам. Это может произойти, если числа отличаются друг от друга в 10 в степени *x* раз, где *x* > 6.

Например: 100 000 000 + 1 = 100 000 000

Формат для строк

Строка – это последовательность символов, причем каждый символ хранится как байт. Первый байт строки определяет ее длину, т.е. количество содержащихся в ней символов. На рис. 4–9 показан формат строки. Строка может включать в себя от 0 до 254 символов, плюс байт, содержащий информацию о длине, таким образом, максимальная длина строки равна 255 байтам.

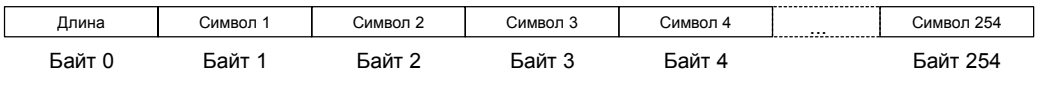


Рис. 4–9. Формат строк

Задание констант для команд S7–200

Во многих командах для S7–200 можно использовать константы. Константы могут быть байтами, словами или двойными словами. S7–200 хранит все константы в виде двоичных чисел, которые могут быть представлены в десятичном, шестнадцатеричном формате, в формате ASCII или в формате вещественных чисел (чисел с плавающей точкой). См. таблицу 4–2.

Таблица 4–2. Представление постоянных величин

Представление	Формат	Пример
Десятичное	[десятичное значение]	20047
Шестнадцатеричное	16#[шестнадцатеричное значение]	16#4E4F
Двоичное	2#[двоичное число]	2#1010_0101_1010_0101
ASCII	'[текст ASCII]'	Текст в одинарных кавычках.'
Вещественное	ANSI/IEEE 754-1985	+1.175495E-38 (положительное) –1.175495E-38 (отрицательное)

Совет
У CPU S7–200 нельзя указывать конкретные типы данных (когда вы, например, хотите указать, что константа должна быть сохранена как целое число (16 битов), целое число со знаком или двойное целое (32 бита)). Например, команда сложения может использовать значение, хранящееся в VW100, как целое число со знаком, а команда "Исключающее ИЛИ" то же самое значение в VW100 может использовать как двоичное значение без знака.

Адресация встроенных входов/выходов и входов/выходов модулей расширения

Встроенные входы и выходы центрального устройства (CPU) имеют фиксированные адреса. Вы можете добавить входы и выходы к CPU S7–200, подключив с правой стороны CPU модули расширения. Адреса входов и выходов на модуле расширения определяются видом входов и выходов, а у нескольких модулей одного типа также их расположением. Например, модуль вывода не влияет на адреса модуля ввода и наоборот. Адреса входов и выходов аналоговых и цифровых модулей также не зависят друг от друга.

Совет

Для цифровых модулей расширения в образе процесса предусмотрены участки по восемь битов (одному байту) каждый. Если в модуле не для каждого бита зарезервированного байта имеется физический вход или выход, то свободные биты теряются и не могут быть поставлены в соответствие следующим модулям расширения этого CPU. У модулей ввода свободные биты в зарезервированных байтах в каждом цикле обновления устанавливаются в ноль.

Входы и выходы аналоговых модулей расширения всегда назначаются парами. Если в модуле не для каждого из этих входов и выходов имеется физический вход или выход, то эти входы и выходы теряются и не могут быть поставлены в соответствие следующим модулям расширения.

На рис. 4–10 показан пример нумерации входов и выходов для конкретной конфигурации аппаратуры. Пропуски в адресации (показаны серым курсивом) не могут использоваться вашей программой.

Рис. 4–10. Пример адресов встроенных входов/выходов и входов/выходов модулей расширения (CPU 224)

Косвенная адресация областей памяти S7–200 с помощью указателей

Косвенная адресация использует указатель для доступа к данным в памяти. Указатели – это ячейки памяти, имеющие размер двойного слова, которые содержат адрес другой ячейки памяти. В качестве указателей можно использовать только ячейки памяти переменных и локальных данных или аккумуляторные регистры (AC1, AC2 или AC3). Для создания указателя необходимо использовать команду "Переместить двойное слово". Эта команда передает адрес косвенно адресованной ячейки памяти в ячейку указателя. Указатели могут также передаваться в подпрограмму в качестве параметров.

S7–200 дает возможность использования указателей для косвенной адресации следующих областей памяти: I, Q, V, M, S, T (только текущее значение) и C (только текущее значение). Косвенную адресацию нельзя использовать для обращения к отдельному биту или к областям памяти AI, AQ, HC, SM или L.

Если вы хотите косвенно обратиться к данным, расположенным по некоторому адресу в памяти, вы можете создать указатель на этот адрес, введя амперсанд (&) и соответствующий адрес. Входному операнду команды должен предшествовать амперсанд (&), чтобы указать на необходимость перемещения в ячейку, обозначенную в выходном операнде команды (указателе), адреса ячейки памяти, а не ее содержимого.

Ввод астериска (*) перед операндом команды указывает, что этот операнд является указателем. Как показано на рис. 4–11, ввод *AC1 указывает, что AC1 является указателем на слово, на которое ссылается команда "Переместить слово" (MOVW). В этом примере значения, хранящиеся в VB200 и VB201, перемещаются в аккумулятор AC0.

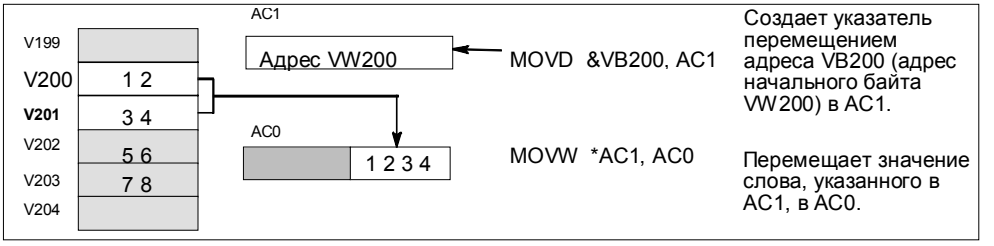


Рис. 4–11. Создание и использование указателя

Как показано на рис. 4–12, вы можете изменить значение указателя. Так как указатели имеют размер 32 бита, то для изменения значений указателей используйте операции над двойными словами. Для изменения значений указателей могут использоваться такие простые математические операции, как сложение или инкрементирование.

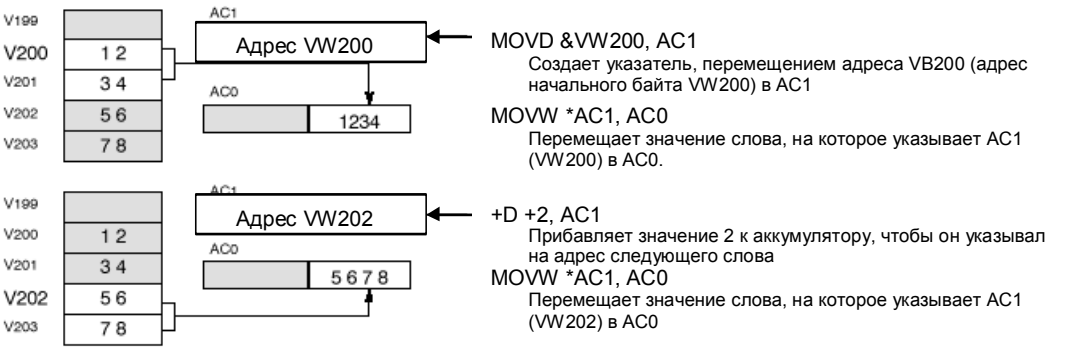
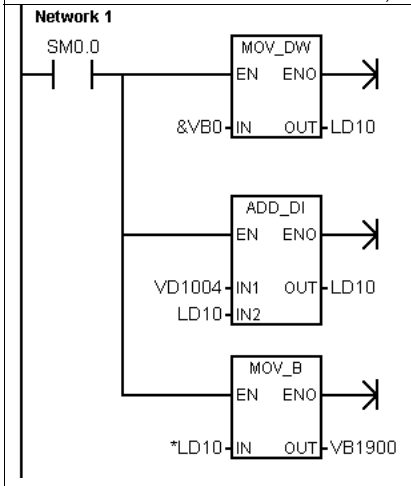


Рис. 4–12. Изменение указателя

Совет
Не забывайте указывать длину данных, к которым вы хотите обратиться: для обращения к байту увеличьте значение указателя на 1; для обращения к слову или текущему значению таймера или счетчика, увеличьте значение указателя на 2, для обращения к двойному слову увеличьте значение указателя на 4.

Пример программы для обращения к данным в памяти переменных с использованием смещения

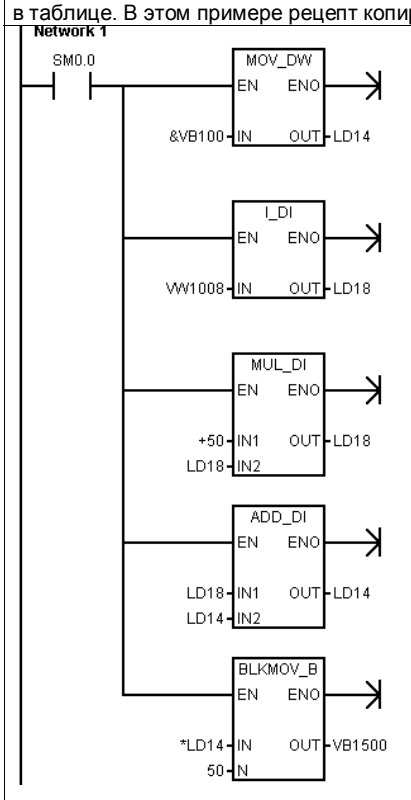
В этом примере используется LD10 как указатель на адрес VB0. Затем вы увеличиваете указатель на величину смещения, хранящуюся в VD1004. Теперь LD10 указывает на другой адрес в памяти переменных (VB0 + смещение). Значение, хранящееся в памяти переменных по адресу, на который указывает LD10, копируется в VB1900. Изменяя значение в VD1004, вы можете обратиться к любому адресу в памяти переменных.




```
Сегмент 1  //Чтение значения из произвольного адреса VB
           //с помощью смещения:
           //1. Загрузить в указатель начальный адрес памяти
           //   переменных.
           //2. Прибавить к указателю величину смещения.
           //3. Скопировать значение из адреса в памяти
           //   переменных (смещение) в VB1900.
LD     SM0.0
MOVD   &VB0, LD10
+D     VD1004, LD10
MOVB   *LD10, VB1900
```

Пример программы для обращения к данным в таблице с использованием смещения

Этот пример использует LD14 как указатель на рецепт, хранящийся в таблице рецептов, которая начинается с VB100. В этом примере VW1008 хранит индекс места конкретного рецепта в таблице. Если каждый рецепт в таблице имеет длину 50 байтов, умножьте индекс на 50, чтобы получить смещение для начального адреса конкретного рецепта. Добавив смещение к указателю, вы можете получить доступ к каждому отдельному рецепту в таблице. В этом примере рецепт копируется в 50 байтов, которые начинаются с VB1500.




```
Сегмент 1  //Передача рецепта из таблицы с рецептами:
           // - каждый рецепт имеет длину 50 байтов.
           // - Индексный параметр (VW1008) идентифицирует
           //   рецепт, подлежащий загрузке.
           //
           //1. Создание указателя на начальный адрес таблицы
           //   рецептов.
           //2. Преобразование индекса рецепта в значение
           //   двойного слова.
           //3. Умножение смещения для учета длины рецепта.
           //4. Прибавление измененного смещения к указателю.
           //5. Передача выбранного рецепта в ячейки с VB1500 по
           //   VB1549.
LD     SM0.0
MOVD   &VB100, LD14
ITD    VW1008, LD18
*D     +50, LD18
+D     LD18, LD14
BMB    *LD14, VB1500, 50
```

Как в S7–200 производится сохранение и извлечение данных

S7–200 предоставляет несколько методов, гарантирующих, что ваша программа, данные программы и конфигурационные данные вашего S7–200 будут сохраняться надлежащим образом.

S7–200 предоставляет в распоряжение конденсатор большой емкости, который поддерживает целостность ОЗУ после отключения питания. В зависимости от исполнения S7–200, этот конденсатор может поддерживать ОЗУ в течение нескольких дней.

S7–200 предоставляет в распоряжение ЭСППЗУ для постоянного, устойчивого к исчезновению напряжения, хранения всей вашей программы, выбранных пользователем областей данных и конфигурационных данных.

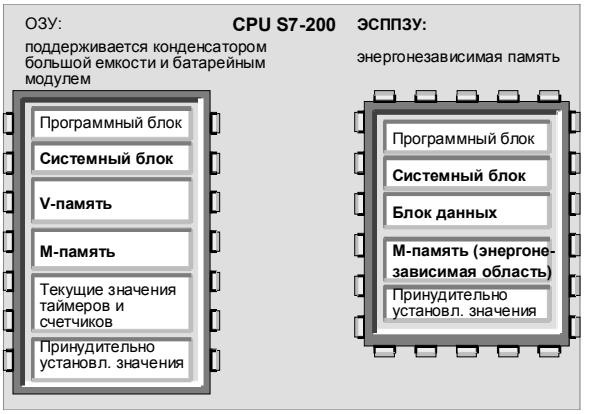


Рис. 4–13. Области памяти CPU S7-200

S7–200 поддерживает необязательный батарейный модуль, который расширяет промежуток времени, в течение которого ОЗУ может сохранять свое содержимое после отключения питания CPU. Батарейный модуль берет на себя буферизацию только после того, как конденсатор разрядился.

Загрузка компонентов вашего проекта в CPU и из CPU

Ваш проект состоит из трех компонентов: программного блока, блока данных (не обязателен) и системного блока (не обязателен).

На рис. 4–14 показано, как проект загружается в S7–200.

При загрузке проекта его компоненты сохраняются в ОЗУ. S7–200 также автоматически копирует программу пользователя, блок данных и системный блок в ЭСППЗУ для энергонезависимого хранения.

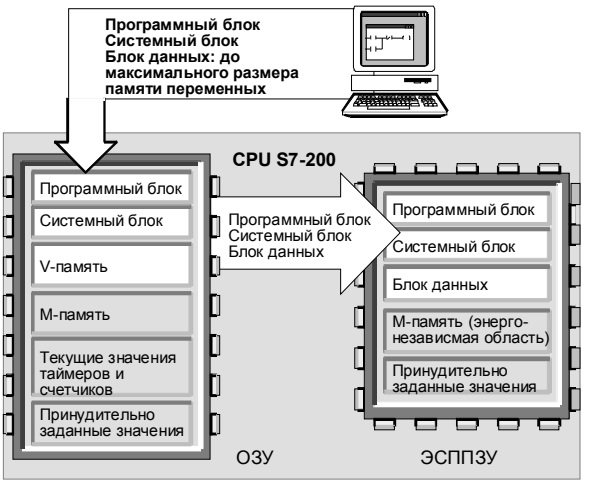


Рис. 4–14. Загрузка проекта в S7–200

На рис. 4–15 показано, как проект загружается из S7–200.

Когда вы загружаете проект в свой компьютер, S7–200 загружает системный блок из ОЗУ, а программный блок и блок данных из ЭСППЗУ.

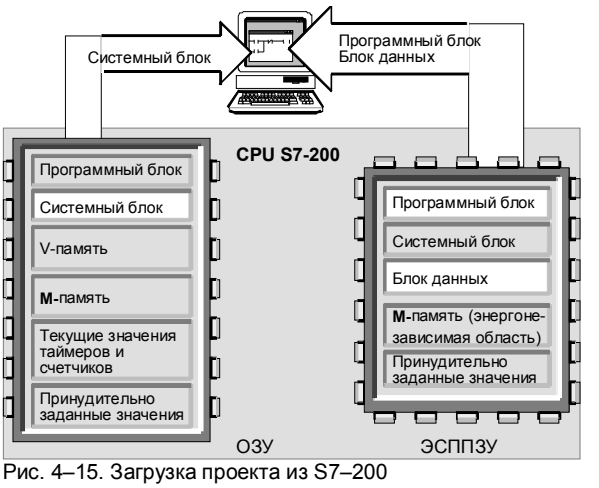


Рис. 4–15. Загрузка проекта из S7–200

Сохранение данных из битовой памяти (M) при потере питания

Если первые 14 байтов битовой памяти (от MB0 до MB13) были определены при конфигурировании как ременантные (сохраняемые), то они сохраняются в ЭСППЗУ, когда S7–200 теряет питание.

Как показано на рис. 4–16, S7–200 передает эти сохраняемые области из M-памяти в ЭСППЗУ.

По умолчанию первые 14 байтов битовой памяти устанавливаются как не сохраняемые. Эта установка деактивизирует процесс сохранения, который обычно происходит при выключении S7–200.

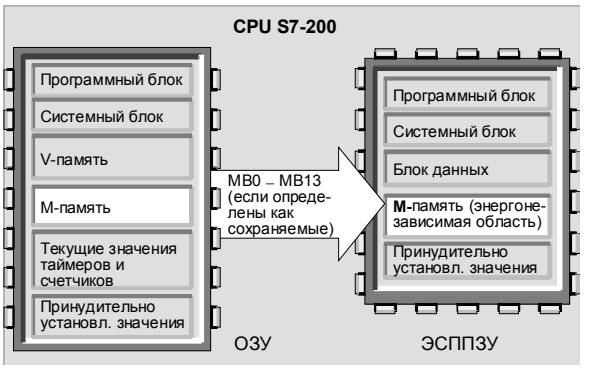


Рис. 4–16. Сохранение битовой памяти при потере питания

Восстановление данных после запуска

При запуске S7–200 восстанавливает программный блок и системный блок из ЭСППЗУ, как показано на рис. 4–17. При запуске S7–200 проверяет ОЗУ, чтобы убедиться, что конденсатор большой емкости успешно выполнил свою задачу по поддержанию данных, хранящихся в ОЗУ. Если содержимое ОЗУ было успешно сохранено, то сохраняемые области ОЗУ остаются неизменными.

Сохраняемые и несохраняемые области памяти переменных восстанавливаются из соответствующего блока данных в ЭСППЗУ. Если содержимое ОЗУ не было сохранено (например, после длительного перерыва в питании), S7–200 очищает ОЗУ (включая сохраняемую и несохраняемую области) и устанавливает бит потери сохраняемых данных (SM0.2) для первого цикла обработки программы, следующего за включением питания, а затем копирует в ОЗУ данные, сохраненные в ЭСППЗУ.

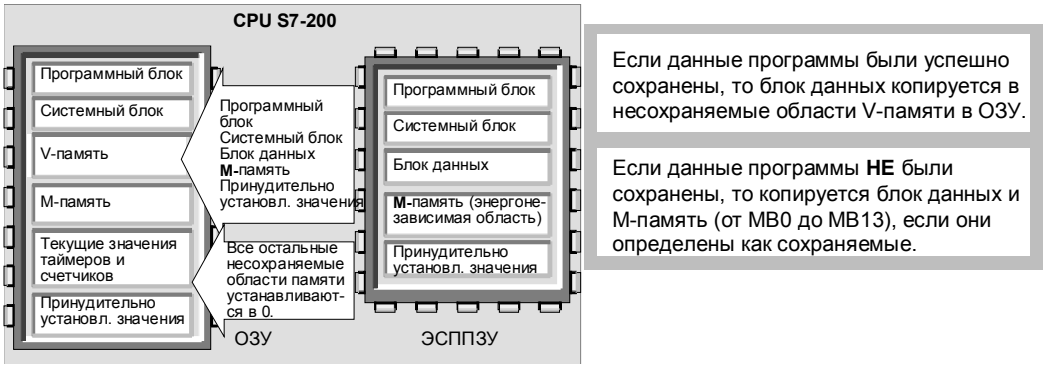


Рис. 4–17. Восстановление данных после запуска

Сохранение программы в модуле памяти

S7–200 поддерживает дополнительный модуль памяти, который предоставляет в распоряжение вашей программе сменное ЭСППЗУ. S7–200 хранит в модуле памяти следующие элементы: программный блок, блок данных, системный блок и принудительно устанавливаемые значения.

Вы можете скопировать свою программу в модуль памяти из ОЗУ только тогда, когда S7–200 получает питание и находится в состоянии STOP, а модуль памяти вставлен. Модуль памяти можно вставлять и вынимать, когда S7–200 включен.

Осторожно
Электростатический разряд может повредить модуль памяти или гнездо на CPU S7–200.

При работе с модулем вы должны находиться на заземленном проводящем полу и/или носить на руке заземленный браслет. Храните модуль в токопроводящем контейнере.

Для установки модуля памяти снимите с CPU S7–200 пластмассовую крышку и вставьте модуль памяти в гнездо. Модуль памяти имеет такую форму, что он может быть вставлен только должным образом.

Копирование вашей программы в модуль памяти

После того как модуль памяти вставлен, используйте для копирования программы следующую процедуру:

1. Переведите CPU S7–200 в состояние STOP.
2. Загрузите программу в S7–200, если она еще не была загружена.
3. Выберите команду меню **PLC** → **Program Memory Cartridge [ПЛК** → **Запрограммировать модуль памяти]**, чтобы скопировать программу в модуль памяти. На рис. 4–18 показаны элементы памяти CPU, которые сохраняются в модуле памяти.
4. Не обязательно: Вытащите модуль памяти и поставьте на место крышку S7–200.

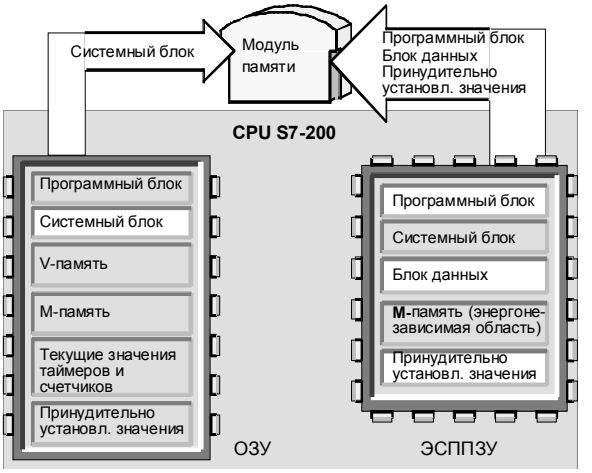


Рис. 4–18. Копирование в модуль памяти

Восстановление программы из модуля памяти

Для передачи программы из модуля памяти в S7–200 вы должны выключить, а затем снова включить питание S7–200 при вставленном модуле памяти.

Примечание
Включение CPU S7–200 с пустым модулем памяти или с модулем памяти, запрограммированным в другой модели CPU S7–200, может вызвать ошибку. Модули памяти, запрограммированные в моделях CPU с меньшими номерами, могут читаться старшими моделями CPU. Противное, однако, неверно. Например, модули памяти, которые были запрограммированы в CPU 221 или CPU 222, могут быть прочитаны CPU 224, но модули памяти, запрограммированные в CPU 224, будут отвергнуты CPU 221 или CPU 222.

Вытащите модуль памяти и снова включите питание S7–200. После этого модуль памяти может быть снова вставлен и перепрограммирован, если необходимо.

Как показано на рис. 4–19, S7–200 выполняет следующие задачи, после того как вы выключили и снова включили питание при вставленном модуле памяти:

1. Если содержимое модуля памяти отличается от содержимого ЭСППЗУ, то S7–200 очищает ОЗУ.
2. S7–200 копирует содержимое модуля памяти в ОЗУ.
3. S7–200 копирует программный блок, системный блок и блок данных в ЭСППЗУ.

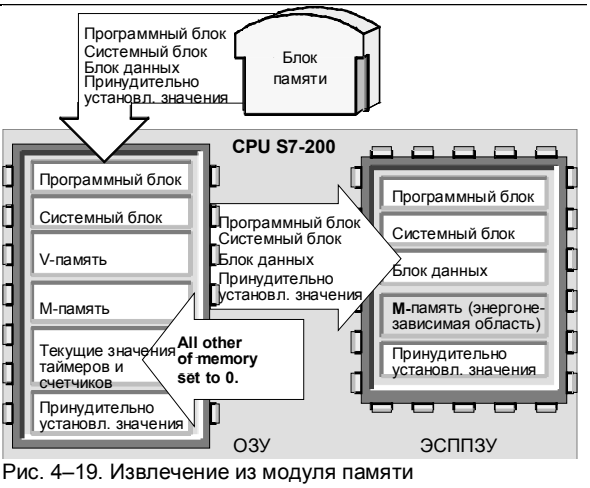


Рис. 4–19. Извлечение из модуля памяти

Установка режима работы CPU S7–200

S7–200 имеет два режима работы: STOP и RUN. Индикаторы состояния на передней панели CPU указывают на текущий режим работы. В состоянии STOP S7–200 не выполняет программы, и вы можете загрузить в CPU программу или конфигурацию CPU. В режиме RUN S7–200 исполняет программу.

- Для изменения режима работы S7–200 снабжен переключателем режимов. С помощью переключателя режимов (он находится под передней крышкой S7–200) вы можете установить режим работы вручную: установка переключателя режимов в STOP прекращает исполнение программы; установка переключателя режимов в RUN запускает исполнение программы, а установка переключателя режимов в режим TERM (терминал) не изменяет режима работы.

  Если питание прерывается, когда переключатель режимов находится в положении STOP или TERM, S7–200 при восстановлении питания автоматически переходит в состояние STOP. Если питание прерывается, когда переключатель режимов находится в положении RUN, S7–200 при восстановлении питания переходит в режим RUN.

- STEP 7-Micro/WIN в режиме online дает возможность изменить режим работы S7–200. Чтобы это программное обеспечение могло управлять режимом работы, вы должны вручную перевести переключатель режимов работы на S7–200 в положение TERM или RUN. Для изменения режима работы вы можете использовать команды меню **PLC > STOP [ПЛК > STOP]** или **PLC > RUN [ПЛК > RUN]** или соответствующие кнопки на панели инструментов.

- Для перевода S7–200 в состояние STOP вы можете использовать в своей программе команду STOP. Это позволяет вам прекратить исполнение своей программы в зависимости от логики обработки программы. Подробную информацию о команде STOP вы найдете в главе 6.

Использование вашей программы для сохранения памяти переменных в ЭСППЗУ

Вы можете сохранить значение (байт, слово или двойное слово), находящееся в любом месте памяти переменных в ЭСППЗУ. Операция сохранения в ЭСППЗУ обычно удлиняет время цикла не более чем на 5 мс. Значение, записанное операцией сохранения, заменяет предыдущее значение, хранящееся в области памяти переменных ЭСППЗУ.

Операция сохранения в ЭСППЗУ не обновляет данные в модуле памяти.

Совет
Так как число операций сохранения в ЭСППЗУ ограничено (минимум 100 000, обычно 1 000 000), вы должны обеспечить, чтобы сохранялись только необходимые значения. В противном случае ЭСППЗУ может износиться, а CPU может выйти из строя. Обычно операции сохранения выполняются при возникновении определенных событий, которые встречаются относительно редко.

Например, если время обработки программы S7–200 составляет 50 мс, а значение сохранялось бы один раз за цикл, то ЭСППЗУ выдержало бы минимум 5 000 секунд, т.е. менее полутора часов. С другой стороны, если значение сохранялось бы один раз в час, то ЭСППЗУ прослужило бы минимум 11 лет.

Копирование V-памяти в ЭСППЗУ

Байт 31 специальной памяти (SMB31) дает S7–200 команду скопировать значение из V-памяти в область памяти переменных ЭСППЗУ. Слово 32 специальной памяти (SMW32) сохраняет адрес копируемой величины. На рис. 4–20 показан формат SMB31 и SMW32.

Выполните следующие шаги, чтобы запрограммировать S7–200 на сохранение или запись конкретного значения в V-памяти:

1. Загрузите адрес значения в V-памяти, которое вы хотите сохранить, в SMW32.
2. Загрузите длину данных в SM31.0 и SM31.1, как показано на рис. 4–20.
3. Установите SM31.7 в 1.

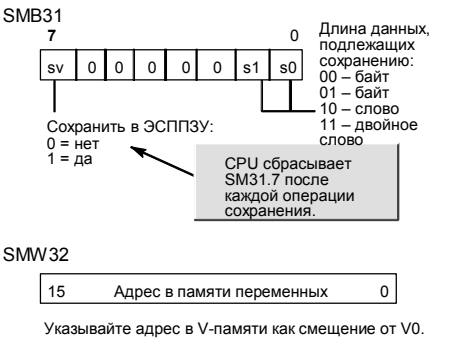


Рис. 4–20. SMB31 и SMW32

В конце каждого цикла выполнения программы S7–200 проверяет SM31.7; если SM31.7 равен 1, то указанное значение сохраняется в ЭСППЗУ. Операция завершается, когда S7–200 сбрасывает SM31.7 в 0.

Не изменяйте значение в V-памяти, пока операция сохранения не будет завершена.

Пример программы: Копирование V-памяти в ЭСППЗУ

Этот пример передает VB100 в ЭСППЗУ. При нарастающем фронте на I0.0, если в это время не происходит другого переноса, происходит загрузка адреса места в памяти переменных, подлежащего передаче, в SMW32. Выбирается длина подлежащей передаче памяти переменных (1 = байт, 2 = слово, 3 = двойное слово или вещественное число). Затем устанавливается SM31.7, чтобы S7–200 передал данные в конце цикла.

По окончании передачи S7–200 автоматически сбрасывает SM31.7 в 0.


```
Сегмент 1      //Передать ячейку памяти
               // переменных (VB100) в ЭСППЗУ
LD      I0.0
EU
AN      SM31.7
MOVW    +100, SMW32
MOVB    1, SMB31
S       SM31.7, 1
```

Функции S7–200

S7–200 предоставляет в распоряжение различные специальные функции, с помощью которых вы можете оптимально настроить S7–200 на свое приложение.

Программа S7–200 может непосредственно производить чтение и запись входов и выходов

Набор команд S7–200 содержит операции непосредственного чтения и записи физических входов/выходов. С помощью этих операций для прямого управления входами и выходами вы можете непосредственно обратиться к входу или выходу, хотя обычно источником или целью обращения к входам и выходам являются образы процесса.

При непосредственном обращении к входу соответствующая ячейка во входном регистре образа процесса не изменяется. При непосредственном обращении к выходу одновременно обновляется соответствующая ячейка в выходном регистре образа процесса.

Совет
S7–200 обрабатывает значения на аналоговых входах как непосредственные данные, если вы не активизировали фильтр на аналоговом входе. При записи значения на аналоговый выход, этот выход обновляется немедленно.

Обычно выгоднее работать с образами процесса и не обращаться во время обработки программы непосредственно к выходам и входам. Есть три существенных причины для использования образов процесса:

- В начале цикла система опрашивает входы. Благодаря этому значения этих входов на время обработки программы синхронизируются и замораживаются. Выходы обновляются после обработки программы через образ процесса. Это обеспечивает стабилизирующее воздействие на систему.
- Ваша программа может обратиться к образу процесса значительно быстрее, чем непосредственно к входам и выходам. Это ускоряет обработку программы.
- Входы и выходы являются битовыми объектами, к которым нужно обращаться в битовом или байтовом формате. Однако к образам процесса можно обращаться в формате бита, байта, слова или двойного слова. Поэтому образы процесса обеспечивают дополнительную гибкость.

Программа S7–200 может прерывать цикл

Если вы используете прерывания, то программы обработки прерываний, которые ставятся в соответствие прерывающим событиям, сохраняются как часть основной программы. Однако они исполняются не как составная часть нормального цикла, а только тогда, когда происходит прерывающее событие (оно возможно в любом месте цикла).

Прерывания обслуживаются S7–200 в последовательности их появления с учетом соответствующих приоритетов. Подробную информацию о командах прерывания вы найдете в главе 6.

S7–200 позволяет выделить время для обработки коммуникационных задач

При проектировании вы можете установить долю времени цикла (в процентах), которая планируется для обработки коммуникационных заданий, которые связаны с процессами компиляции в режиме RUN или с состоянием выполнения. (Редактирование в режиме RUN и состояние исполнения – это возможности, предоставляемые пакетом STEP 7-Micro/WIN для облегчения отладки вашей программы.) Увеличение доли времени, отводимой на коммуникационные задачи, увеличивает время цикла, вследствие чего ваш процесс управления осуществляется медленнее.

По умолчанию доля времени цикла, отводимая на коммуникационные задания, составляет 10%. Эта установка является разумным компромиссом для обработки компиляций и состояния, минимизируя влияние на процесс управления. Вы можете настраивать это значение шагами по 5% максимум до 50%. Если вы хотите установить время для обмена данными в фоновом режиме, действуйте следующим образом:

1. Выберите команду меню **View > Component > System Block [Вид > Компонент > Системный блок]** и щелкните на закладке Background Time [Фоновое время].
2. Измените свойства для фонового времени обмена данными и щелкните на OK.
3. Загрузите измененный системный блок в S7–200.

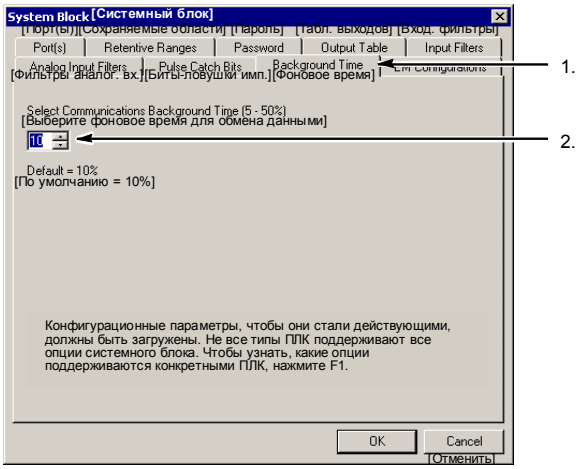


Рис. 4–21. Фоновое время для обмена данными

S7–200 дает возможность устанавливать состояния цифровых выходов в режиме STOP

С помощью таблицы выходов S7–200 вы можете установить сигнальные состояния цифровых выходов при переходе в режим STOP на определенные значения, или вы можете "заморозить" выходы точно в том состоянии, в котором они находились перед переходом в STOP. Таблица выходов – это часть системного блока, которая загружается и сохраняется в S7–200 и действительна только для цифровых выходов.

1. Выберите команду меню **View > Component > System Block [Вид > Компонент > Системный блок]** и щелкните на закладке Output Table [Таблица выходов].
2. Для замораживания выходов в их последнем состоянии активизируйте триггерную кнопку Freeze Outputs [Заморозить выходы].
3. Для копирования табличных значений в выходы введите эти значения в таблицу выходов, щелкая на триггерной кнопке для каждого выходного бита, который вы хотите установить в 1 после перехода из RUN в STOP. (По умолчанию все значения в таблице равны нулю.)
4. Подтвердите введенные значения, щелкнув на OK.
5. Загрузите измененный системный блок в S7–200.

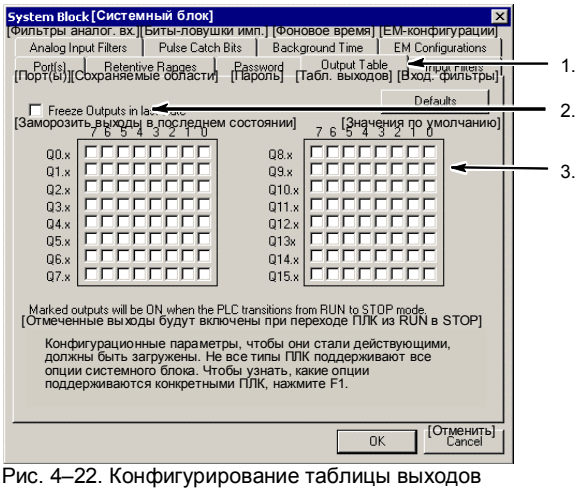


Рис. 4–22. Конфигурирование таблицы выходов

S7–200 позволяет определить память, которая сохраняется при потере питания

Вы можете определить в качестве сохраняемых до шести областей и выбрать области памяти, которые вы хотели бы буферизовать при потере питания. Вы можете определить диапазоны адресов, которые должны быть сохраняемыми, в следующих областях памяти: V, M, C и T. У таймеров могут быть буферизованы только сохраняемые таймеры (TONR). По умолчанию первые 14 байтов битовой (M) памяти не сохраняются.

У таймеров и счетчиков могут быть буферизованы только текущие значения: биты таймеров и счетчиков не сохраняются.

Совет
Если вы определите диапазон от MB0 до MB13 в качестве сохраняемого, то активизируется специальная функция, которая при потере питания автоматически сохраняет эти ячейки памяти в ЭСППЗУ.

Для определения сохраняемой памяти:

1. Выберите команду меню **View > Component > System Block [Вид > Компонент > Системный блок]** и щелкните на закладке Retentive Ranges [Сохраняемые области].
2. Выберите области в памяти, которые должны быть буферизованы при потере питания, и щелкните на OK.
3. Загрузите измененный системный блок в S7–200.

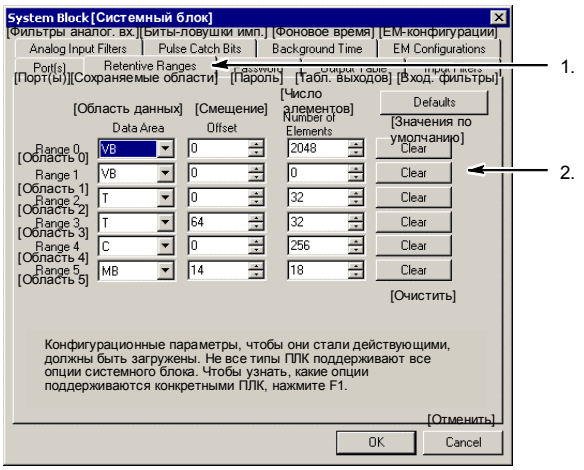


Рис. 4–23. Сохраняемая память

S7–200 дает возможность фильтровать цифровые входы

S7–200 позволяет выбрать входной фильтр, который определяет время задержки (выбираемое в пределах от 0,2 мс до 12,8 мс) для всех или некоторых встроенных цифровых входов. Эта задержка помогает отфильтровать шум во входной проводке, который может вызвать непреднамеренные изменения состояний входов.

Входной фильтр является частью системного блока, которая загружается и хранится в S7–200. По умолчанию время фильтра равно 6,4 мс. Как показано на рис. 4–24, каждая заданная задержка действительна для группы входов.

Для конфигурирования времен задержки для входного фильтра:

1. Выберите команду меню **View > Component > System Block [Вид > Компонент > Системный блок]** и щелкните на закладке Input Filters [Входные фильтры].
2. Введите величину задержки для каждой группы входов и щелкните на OK .
3. Загрузите измененный системный блок в S7–200.

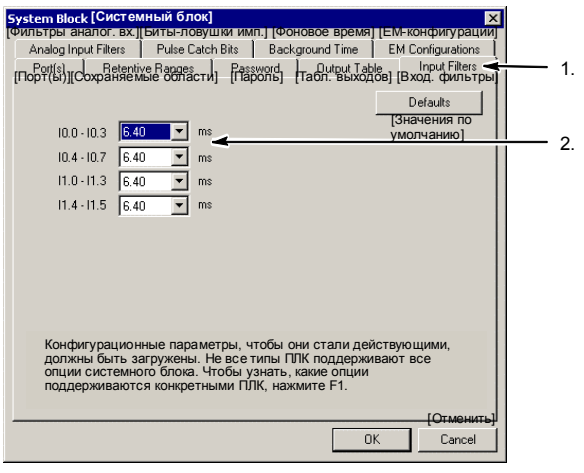


Рис. 4–24. Конфигурирование входных фильтров

Совет
Фильтр цифровых входов оказывает также влияние на входную величину в командах чтения, входных прерываний и ловушек импульсов. В зависимости от настройки фильтра это может привести к тому, что ваша программа может пропустить прерывающее событие или импульс. Скоростные счетчики подсчитывают события на входах без фильтров.

S7–200 дает возможность фильтровать аналоговые входы

У S7–200 вы можете установить программный фильтр для отдельных аналоговых входов. Отфильтрованное значение является средним значением заранее установленного количества опросов аналоговых входов. Параметры фильтра (количество опросов и зона нечувствительности) одинаковы для всех аналоговых входов, для которых фильтр активизирован.

Фильтр обладает свойством быстрой реакции, что обеспечивает быстрое воздействие больших изменений на значение фильтра. Фильтр обеспечивает реакцию на последнее значение на аналоговом входе, как на ступенчатое воздействие, если изменение на этом входе превышает определенную величину. Это изменение, называемое зоной нечувствительности, задается в отсчетах цифрового значения аналогового входа.

По умолчанию фильтр активизирован для всех аналоговых входов.

1. Выберите команду меню **View > Component > System Block [Вид > Компонент > Системный блок]** и щелкните на закладке Analog Input Filters [Фильтры аналоговых входов].
2. Выберите аналоговые входы, которые вы хотите фильтровать, количество опросов и зону нечувствительности.
3. Щелкните на OK .
4. Загрузите измененный системный блок в S7–200.

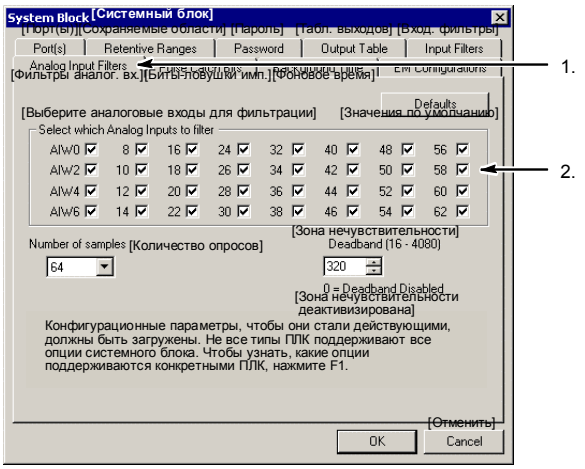


Рис. 4–25. Фильтр аналоговых входов

Совет
Не используйте аналоговый фильтр с модулями, которые передают цифровую информацию или аварийные сигналы в аналоговых словах. Всегда выключайте аналоговый фильтр для модулей с RTD, термопарой и главного модуля AS–интерфейса.

S7–200 дает возможность регистрировать короткие импульсы

S7–200 имеет в своем распоряжении функцию "Захват импульса", которая может быть использована для всех или некоторых встроенных цифровых входов. Функция " Захват импульса" дает возможность регистрировать импульсы большой или малой амплитуды, имеющие столь малую продолжительность, что они легко могут быть пропущены модулем S7–200, который считывает цифровые входы в начале цикла. Если функция " Захват импульса" активизирована для некоторого входа, то изменение сигнала на этом входе фиксируется и удерживается, пока не произойдет обновление данных в следующем цикле. Это гарантирует, что импульс, длящийся очень короткий интервал времени, будет зарегистрирован и сохранен, пока S7–200 не прочтет входы.

Функцию " Захват импульса" можно активизировать для любого встроенного цифрового входа.

Чтобы вызвать диалоговое окно для конфигурирования захвата импульса:

1. Выберите команду меню **View > Component > System Block [Вид > Компонент > Системный блок]** и щелкните на закладке Pulse Catch Bits [Биты-ловушки импульсов].
2. Активизируйте желаемую триггерную кнопку и щелкните на OK .
3. Загрузите измененный системный блок в S7–200.

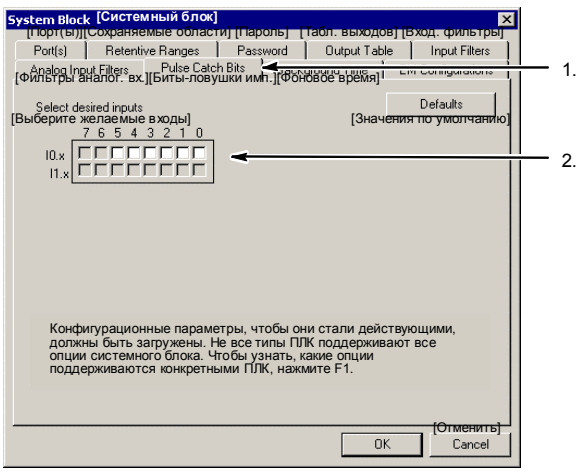


Рис. 4–26. Захват импульсов

На рис. 4–27 показано, как действует S7–200 с активизированным и деактивизированным захватом импульсов.

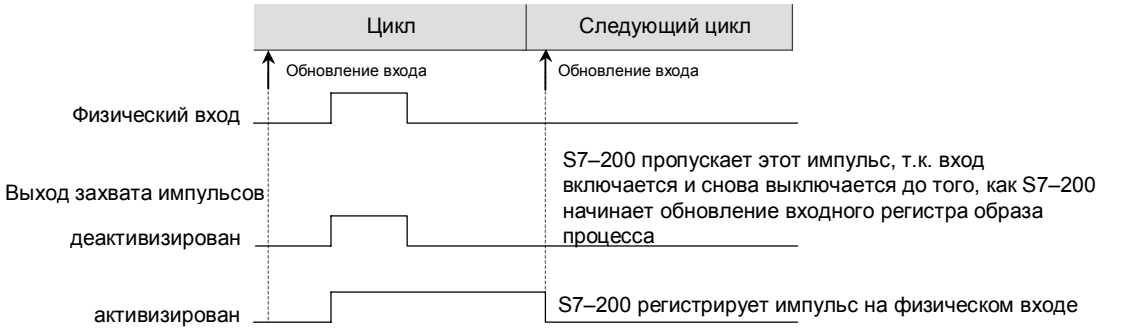


Рис. 4–27. Функционирование S7–200 с активизированным и деактивизированным захватом импульсов

Так как функция захвата импульсов работает на входе после того, как сигнал прошел через входной фильтр, вы должны так настроить время входного фильтра, чтобы импульс не был удален фильтром. На рис. 4–28 дано схематическое представление цепи цифрового входа.

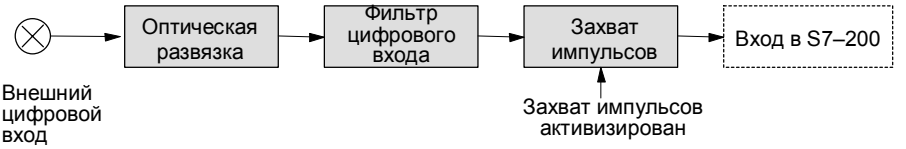


Рис. 4–28. Цепь цифрового входа

На рис. 4–29 показана реакция активизированного захвата импульсов на различные входные условия. Если в данном цикле имеется более одного импульса, то регистрируется только первый из них. При нескольких импульсах в одном цикле вам следует использовать прерывающие события для нарастающего и падающего фронтов. (Перечисление прерывающих событий вы найдете в таблице 6–44.)

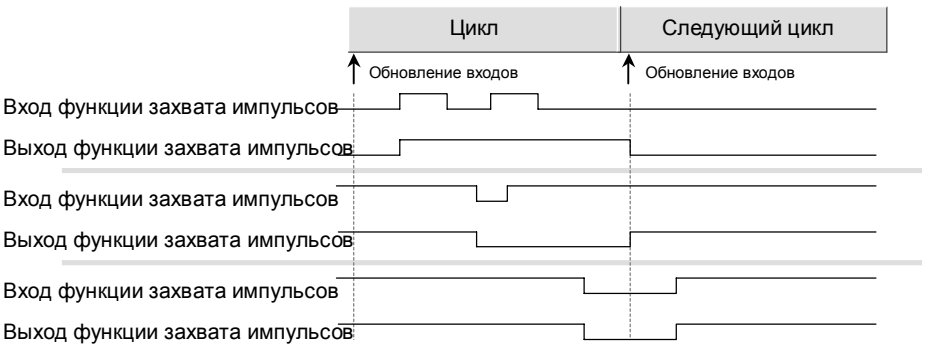


Рис. 4–29. Реакции функции захвата импульсов на различные входные условия

S7–200 предоставляет защиту с помощью пароля

Все модели S7–200 предоставляют защиту с помощью пароля для ограничения доступа к определенным функциям.

Благодаря установке пароля только уполномоченные лица имеют доступ к определенным функциям и памяти: без пароля возможен неограниченный доступ к S7–200. При наличии парольной защиты S7–200 ограничивает доступ к функциям в соответствии с конфигурацией пароля.

Пароль не чувствителен к регистру символов.

Как показано в таблице 4–3, S7–200 предоставляет три уровня ограничения доступа. Каждый уровень предоставляет неограниченный доступ к определенным функциям без ввода пароля. Для всех трех уровней ввод правильного пароля предоставляет доступ ко всем функциям. По умолчанию для S7–200 установлен уровень 1 (без ограничений).

Ввод пароля через сеть не оказывает влияния на парольную защиту S7–200.

Таблица 4–3. Ограничение доступа к S7–200

Функция CPU	Уровень 1	Уровень 2	Уровень 3
Чтение и запись данных пользователя	Доступ разрешен	Доступ разрешен	Доступ разрешен
Запуск, останов и перезапуск CPU			
Чтение и установка часов реального времени			
Загрузка программы пользователя, данных и конфигурации CPU из CPU	Доступ разрешен	Доступ разрешен	Требуется пароль
Загрузка в CPU	Доступ разрешен	Требуется пароль	
Получение состояния выполнения			
Удаление программного блока, блока данных и системного блока			
Принудительное задание данных или исполнение одного или нескольких циклов			
Копирование в модуль памяти			
Запись в выходы в состоянии STOP			

Если один пользователь имеет право доступа к защищенным функциям, то другие пользователи не имеют права доступа к этим функциям. В каждый данный момент времени неограниченный доступ к S7–200 имеет только один пользователь.

Совет
После того как вы ввели пароль, уровень защиты для этого пароля остается действительным в течение максимум одной минуты после отсоединения устройства программирования от S7–200. Всегда выходите из STEP 7–Micro/WIN перед отсоединением кабеля, чтобы другой пользователь не мог получить доступа к привилегиям этого устройства программирования.

Установка пароля для S7–200

Диалоговое окно System Block [Системный блок] (рис. 4–30) позволяет установить пароль для S7–200:

1. Выберите команду меню **View > Component > System Block [Вид > Компонент > Системный блок]** для отображения диалогового окна System Block [Системный блок] и щелкните на закладке Password [Пароль].
2. Выберите желаемый уровень защиты для S7–200.
3. Введите пароль и подтвердите его.
4. Щелкните на OK .
5. Загрузите измененный системный блок в S7–200.

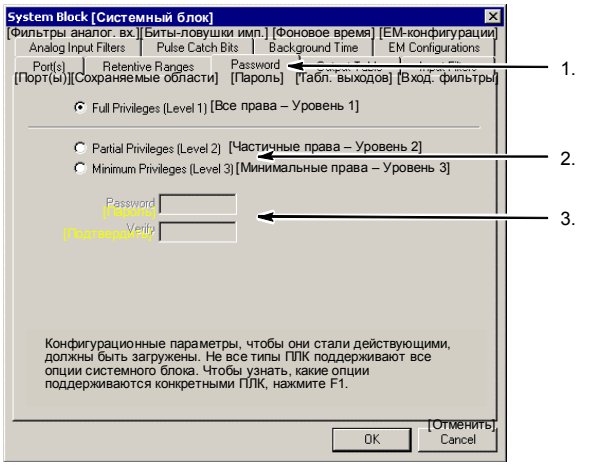


Рис. 4–30. Создание пароля

Последовательность действий при утере пароля

Если вы забыли пароль, то вы должны очистить память S7–200 и перезагрузить свою программу. Очистка памяти переводит S7–200 в режим STOP и восстанавливает в S7–200 значения заводской настройки, за исключением адреса, скорости передачи и часов реального времени. Для стирания программы в S7–200:

1. Выберите команду меню **PLC > Clear [ПЛК → Очистить...]**, чтобы отобразить диалоговое окно Clear [Очистка].
2. Выделите все три блока и подтвердите ваше действие щелчком на кнопке OK.
3. Если пароль был создан, то STEP 7-Micro/WIN отображает диалоговое окно, в котором запрашивается пароль доступа. Для стирания пароля введите в этом диалоговом окне CLEARPLC, чтобы продолжить операцию общего стирания (Clear All). (Пароль CLEARPLC не чувствителен к регистру шрифта.)

При общем стирании программа в модуле памяти сохраняется. Так как модуль памяти наряду с программой хранит пароль, вы должны перепрограммировать также модуль памяти, чтобы удалить потерянный пароль.

Предупреждение

Очистка памяти S7–200 вызывает выключение выходов (или “замораживание” на определенном уровне в случае аналогового выхода).

Если во время очистки памяти S7–200 соединен с оборудованием, то изменения состояний выходов могут передаваться этому 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 гибель или тяжкие телесные повреждения персонала и/или повреждение оборудования.

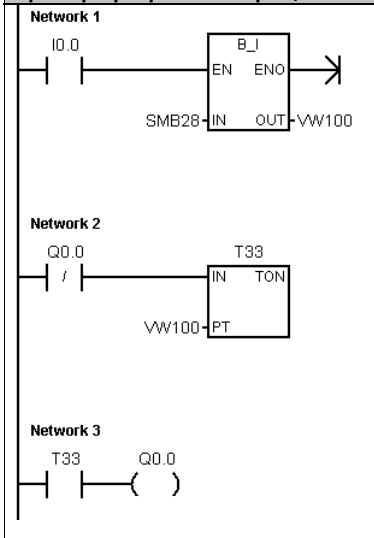
Всегда соблюдайте соответствующие меры безопасности и перед очисткой памяти S7–200 обеспечьте, что ваш процесс находится в безопасном состоянии.

S7–200 имеет в своем распоряжении аналоговые потенциометры

Аналоговые потенциометры для настройки находятся под передней крышкой модуля. Вы можете настраивать эти потенциометры для увеличения или уменьшения значений, хранящихся в байтах в специальной памяти (SMB). Эти защищенные от записи величины могут использоваться программой для реализации ряда функций, например, актуализация текущего значения таймера или счетчика, ввод или изменение предустановленных значений или установка граничных значений. Для настройки нужна маленькая отвертка: поверните потенциометр по часовой стрелке (направо) для увеличения значения и против часовой стрелки (налево) для уменьшения значения.

SMB28 хранит цифровое значение, представляющее настройку аналогового потенциометра 0. SMB29 хранит цифровое значение, представляющее настройку аналогового потенциометра 1. Аналоговый потенциометр имеет номинальный диапазон от 0 до 255 и повторяемость ± 2 отсчета.

Пример программы обращения к величине, введенной с помощью аналогового потенциометра




```
Network 1     //Прочитать аналоговый потенциометр 0 (SMB28).
              //Сохранить значение как целое в слове VW100.
LD    I0.0
BTI   SMB28, VW100

Network 2     //Использовать целое значение (VW100) в
              //качестве уставки для таймера.
LDN   Q0.0
TON   T33, VW100

Network 3     //Включить Q0.0, когда T33 достигнет величины
              //уставки.
LD    T33
=     Q0.0
```

У S7–200 имеются скоростные входы и выходы

Скоростные счетчики

S7–200 предоставляют в распоряжение встроенные скоростные счетчики, которые считают быстро протекающие внешние события без ухудшения производительности S7–200. Скорости, поддерживаемые вашей моделью CPU, вы найдете в Приложении A. У каждого счетчика имеются входы, предназначенные для синхронизации, управления направлением, сброса и запуска, где эти функции поддерживаются. Вы можете варьировать скорость счета установкой различных A/B-счетчиков. За дополнительной информацией об использовании скоростных счетчиков обратитесь к главе 6.

Скоростные импульсные выходы

S7–200 поддерживает скоростные импульсные выходы, причем выходы Q0.0 и Q0.1 могут генерировать последовательности скоростных импульсов (PTO) или выполнять управление с помощью широтно-импульсной модуляции (PWM).

Функция «Последовательность скоростных импульсов» дает выход в виде прямоугольных импульсов (с относительной длительностью 50 %) для заданного количества импульсов и заданного времени цикла. Количество импульсов может быть задано в диапазоне от 1 до 4 294 967 295. Время цикла может быть задано в микро- или миллисекундах в пределах от 50 мкс до 65 535 мкс или от 2 мс до 65 535 мс. Функция «Последовательность скоростных импульсов» (PTO) может быть запрограммирована так, чтобы реализовать одну последовательность импульсов или конфигурацию, состоящую из нескольких последовательностей импульсов. Например, для управления шаговым двигателем вы можете использовать конфигурацию импульсов, состоящую из линейно нарастающего участка, рабочего участка и линейно убывающего участка, или более сложные последовательности. Конфигурация импульсов может включать в себя до 255 сегментов с участками, соответствующими разгону, работе или замедлению.

Функция «Широтно-импульсная модуляция» обеспечивает фиксированное время цикла с переменной относительной длительностью импульсов. Время цикла и ширина импульсов могут быть заданы в микро- или миллисекундах. Время цикла имеет диапазон от 50 мкс до 65 535 мкс или от 2 мс до 65 535 мс. Ширина импульсов имеет диапазон от 0 мкс до 65 535 мкс или от 0 мс до 65 535 мс. Когда ширина импульса равна времени цикла, относительная длительность импульсов равна 100 процентам, и выход включен постоянно. Когда ширина импульсов равна нулю, относительная длительность импульсов равна 0 процентов, и выход выключен.

За дополнительной информацией о скоростных импульсных выходах обратитесь к главе 6.