Память CPU: типы данных и способы адресации

5

CPU S7–200 предоставляет специализированные области памяти для обеспечения более быстрой и эффективной обработки данных.

Обзор главы

5.1 Прямая адресация областей памяти CPU

CPU S7–200 хранит информацию в различных ячейках памяти, имеющих уникальные адреса. Вы можете явно указать адрес в памяти, к которому вы хотите обратиться. Это позволяет вашей программе иметь прямой доступ к информации.

Использование адреса памяти для доступа к данным

Для обращения к биту в области памяти вы указываете адрес, который включает в себя идентификатор области памяти, адрес байта и номер бита. Рис. 5–1 дает пример обращения к биту (это называется также адресацией “байт.бит”). В этом примере за областью памяти и адресом байта (I = вход (input),а 3 = байт 3) следует точка (“.”), чтобы отделить адрес бита (бит 4).

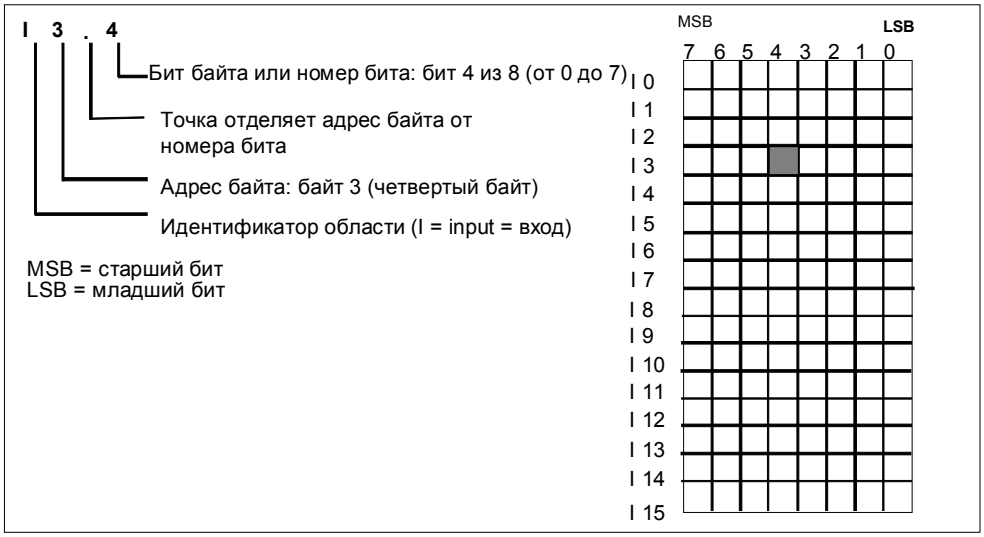

Рис. 5-1. Обращение к биту данных в памяти CPU (адресация байт.бит)

Используя байтовый формат адреса вы можете обращаться к данным в различных областях памяти CPU (V, I, Q, M, S, L и SM) как к байтам, словам или двойным словам. Для доступа к байту, слову или двойному слову в памяти CPU вы должны указать адрес, таким же способом, как и при указании адреса бита. Он включает идентификатор области, обозначение размера данных и начальный байтовый адрес байта, слова или двойного слова, как это показано на рис. 5–2. Доступ к данным в других областях памяти CPU (таким, как T, C, HC и аккумуляторы) производится с помощью формата адреса, включающего идентификатор области и номер элемента.

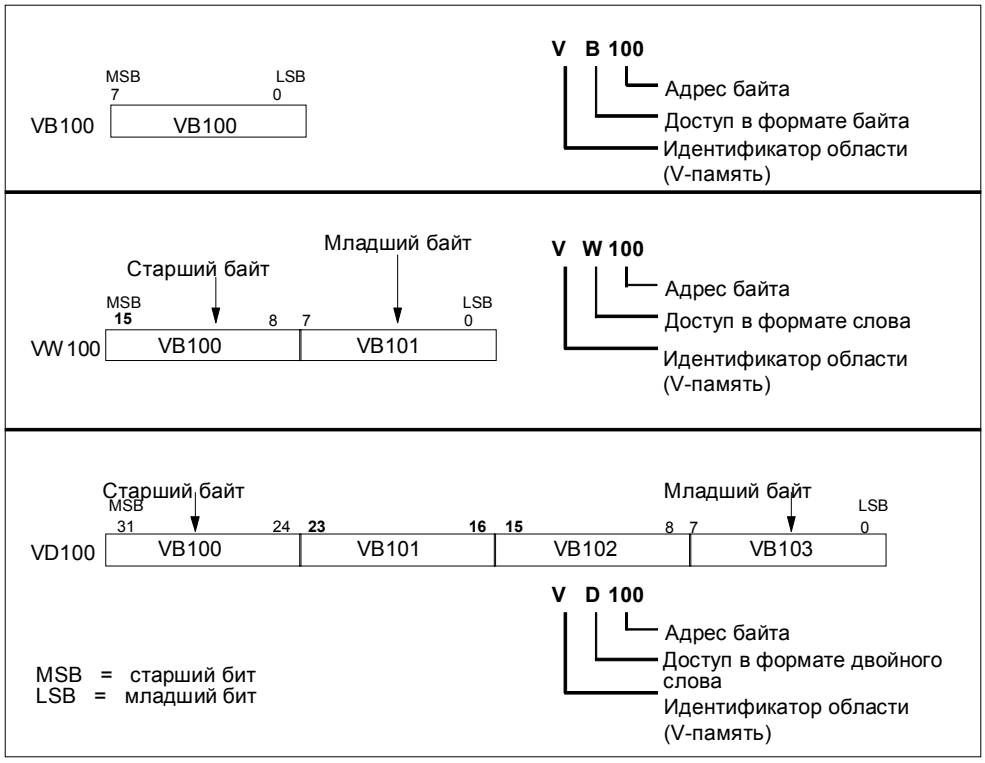

Рис. 5-2. Сравнение доступа к байту, слову и двойному слову по одному и тому же адресу

Представление чисел

В таблице 1–1 показан диапазон числовых значений, которые могут быть представлены различными размерами данных.

Вещественные числа (числа с плавающей точкой) представляются как 32-битные числа одинарной точности в формате: от +1.175495E-38 до +3.402823E+38 (положительные) и от -1.175495E-38 до -3.402823E+38 (отрицательные). Доступ к значениям вещественных чисел производится с помощью двойных слов. За дополнительной информацией о вещественных числах или числах с плавающей точкой обратитесь к стандарту ANSI/IEEE 754-1985.

Таблица 1-1. Обозначения размеров данных и соответствующие диапазоны целых чисел

Размер данных	**Диапазон целых чисел без знака**		**Диапазон целых чисел со знаком**	
	десятичные	**16-ричные**	**десятичные**	**16-ричные**
B (байт): 8–битовое значение	от 0 до 255	от 0 до FF	от -128 до 127	от 80 до 7F
W (слово): 16–битовое значение	от 0 до 65 535	от 0 до FFFF	от –32 768 до 32 767	от 8000 до 7FFF
D (двойное слово): 32–битовое значение	от 0 до 4 294 967 295	от 0 до FFFF FFFF	от –2 147 483 648 до 2 147 483 647	от 8000 0000 до 7FFF FFFF

Адресация регистра входов образа процесса (I)

Как описано в разделе 4.6, CPU опрашивает физические входы в начале каждого цикла и записывает эти значения в регистр входов образа процесса. Вы можете получить доступ к регистру входов образа процесса в битах, байтах, словах или двойных словах.

Формат:

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

Адресация регистра выходов образа процесса (Q)

В конце цикла CPU копирует значения, хранящиеся в регистре выходов образа процесса в физические выходы. Вы можете получить доступ к регистру выходов образа процесса в битах, байтах, словах или двойных словах.

Формат:

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

Адресация области памяти переменных (V)

Вы можете использовать V-память для хранения промежуточных результатов операций, выполняемых в вашей программе управляющей логикой. Вы можете также использовать V-память для хранения других данных, относящихся к вашему процессу или задаче. Вы можете получить доступ к V-памяти в битах, байтах, словах или двойных словах.

Формат:

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

Адресация битовой памяти (меркеров) (M)

Вы можете использовать область битовой памяти (M-память) в качестве управляющих реле для хранения промежуточного состояния операции или другой управляющей информации. Хотя название “область битовой памяти” подразумевает, что информация хранится в единицах длиной в бит, вы можете получить доступ к области битовой памяти не только в битах, но также и в байтах, словах или двойных словах.

Формат:

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

Адресация области памяти реле управления последовательностью операций (S)

Биты реле управления последовательностью операций (S) используются для организации машинных операций или шагов в эквивалентные сегменты программы. Реле управления последовательностью операций делают возможной сегментацию программы управления. Вы можете получить доступ к области S в битах, байтах, словах или двойных словах.

Формат:

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

Адресация битов специальной памяти (SM)

SM-биты предоставляют средства для обмена информацией между CPU и вашей программой. Вы можете использовать эти биты для выбора и управления некоторыми специальными функциями CPU S7–200, такими, как:

- бит, устанавливающийся при первом цикле обработки программы
- бит, переключающийся с определенной частотой
- биты, показывающие состояние арифметических или операционных команд

Дополнительную информацию об SM-битах вы найдете в приложении C. Хотя SM-область основана на битах, вы можете получить обращаться к данным в этой области как к битам, байтам, словам или двойным словам.

Формат:

Бит **SM*[адрес байта].[адрес бита]***
SM0.1

Байт, слово, двойное слово **SM*[размер][начальный адрес байта]***
SMB86

Адресация области локальной памяти (L)

ПЛК S7–200 предоставляет 64 байта локальной памяти (L), из которых 60 могут быть использованы в качестве сверхоперативной памяти или для передачи формальных параметров подпрограммам. Если вы программируете на LAD (КОР) или FBD (FUP), то STEP 7-Micro/WIN 32 резервирует последние 4 байта локальной памяти для собственных целей. Если вы программируете на STL (AWL), то доступны все 64 байта L-памяти, но не рекомендуется использовать последние четыре байта этой памяти.

Локальная память похожа на V-память за одним существенным исключением. V-память имеет глобальную сферу действия, тогда как L-память имеет локальную сферу действия. Термин глобальная сфера действия означает, что одна и та же ячейка памяти доступна из любого программного объекта (главной программы, подпрограмм или программ обработки прерываний). Термин локальная сфера действия означает, что распределение памяти ассоциируется с конкретным программным объектом. ПЛК S7–200 выделяют 64 байта L-памяти для главной программы, 64 байта для каждого уровня вложенности подпрограмм и 64 байта для программ обработки прерываний.

L-память, выделенная для главной программы, недоступна из подпрограмм или программ обработки прерываний. Подпрограмма не может получить доступ к L-памяти, выделенной для главной программы, программы обработки прерывания или другой подпрограммы. Аналогично, программа обработки прерывания не может получить доступ к L-памяти, выделенной для главной программы или подпрограммы.

Выделение L-памяти производится ПЛК S7–200 по принципу «сколько нужно». Это значит, что пока выполняется главная часть программы, выделение L-памяти для подпрограмм и программ обработки прерываний не производится. При возникновении прерывания или вызове подпрограммы выделяется требуемое количество локальной памяти. Новое выделение L-памяти может повторно использовать ячейки L-памяти другой подпрограммы или программы обработки прерывания.

L-память не инициализируется ПЛК при ее выделении и может содержать любое значение. Когда вы передаете формальные параметры в вызов подпрограммы, значения передаваемых параметров помещаются CPU в соответствующие ячейки L-памяти вызываемой подпрограммы. Ячейки L-памяти, которые не получают значения при передаче формальных параметров не будут инициализированы и могут содержать во время выделения памяти любое значение.

Вы можете получить доступ к L-памяти в виде битов, байтов, слов или двойных слов. Вы можете использовать L-память как указатель для косвенной адресации, но вы не можете косвенно обращаться к L-памяти.

Формат:

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

Адресация таймеров (T)

В CPU S7–200 таймеры – это элементы, которые считают приращения времени. Таймеры S7–200 имеют разрешения (приращения базы времени) в 1 мс, 10 мс или 100 мс. С таймером связаны две переменные:

- Текущее значение: это 16–битовое целое число со знаком хранит количество времени, отсчитанное таймером.
- Бит таймера: этот бит устанавливается или сбрасывается как результат сравнения текущего и предустановленного значения. Предустановленное значение вводится как часть команды таймера.

Доступ к обеим переменным производится с помощью адреса (T + номер таймера). К какому из этих элементов, биту таймера или текущему значению, производится обращение, зависит от используемой команды: команды с битовым операндом обращаются к биту таймера, тогда как команды, имеющие в качестве операнда слово, обращаются к текущему значению. Как показано на рис. 5–3, команда «Нормально открытый контакт» обращается к биту таймера, тогда как команда «Переместить слово» (MOV_W) обращается к текущему значению таймера. Для получения дополнительной информации о командах S7–200 обратитесь к главе 9 (для команд SIMATIC) и к главе 10 (для команд IEC 1131-3).

Формат: **T*[номер таймера]*** **T24**

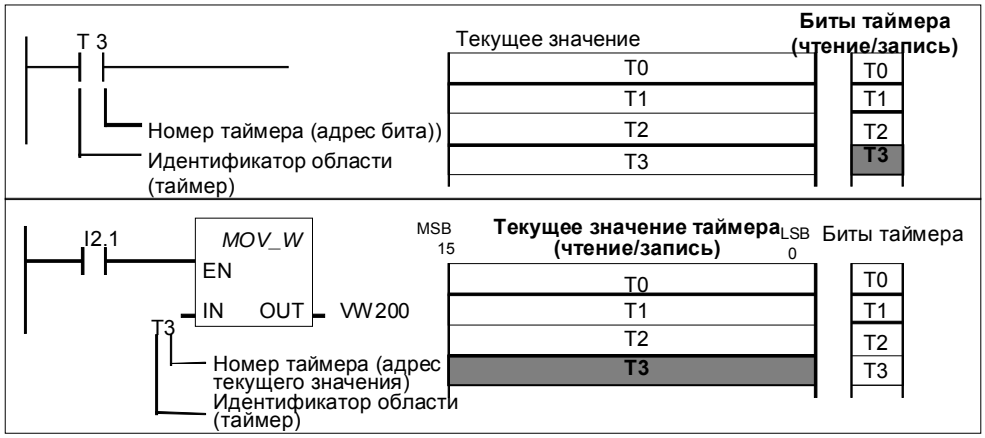

Рис. 5-3. Доступ к данным таймера SIMATIC

Адресация счетчиков (C)

В CPU S7–200 счетчики – это элементы, которые считают каждый нарастающий фронт на входе (входах) счетчика. CPU предоставляет три типа счетчиков: один считает только вперед, один – только назад и один – как вперед, так и назад. Со счетчиком связаны две переменные величины:

- Текущее значение: это 16–битовое целое число со знаком хранит накопленное счетчиком значение
- Бит счетчика: этот бит устанавливается или сбрасывается как результат сравнения текущего и предустановленного значения. Предустановленное значение вводится как часть команды счетчика.

Обращение к обеим переменным производится с помощью адреса счетчика (C + номер счетчика). К какому из этих элементов, биту счетчика или текущему значению, производится обращение, зависит от используемой команды: команды с битовым операндом обращаются к биту счетчика, тогда как команды, имеющие в качестве операнда слово, обращаются к текущему значению. Как показано на рис. 5–4, команда «Нормально открытый контакт» обращается к биту счетчика, тогда как команда «Переместить слово» (MOV_W) обращается к текущему значению счетчика. Для получения дополнительной информации о командах S7–200 обратитесь к главе 9 (для команд SIMATIC) и к главе 10 (для команд IEC 1131-3).

Формат: ***C[номер счетчика]*** **C20**

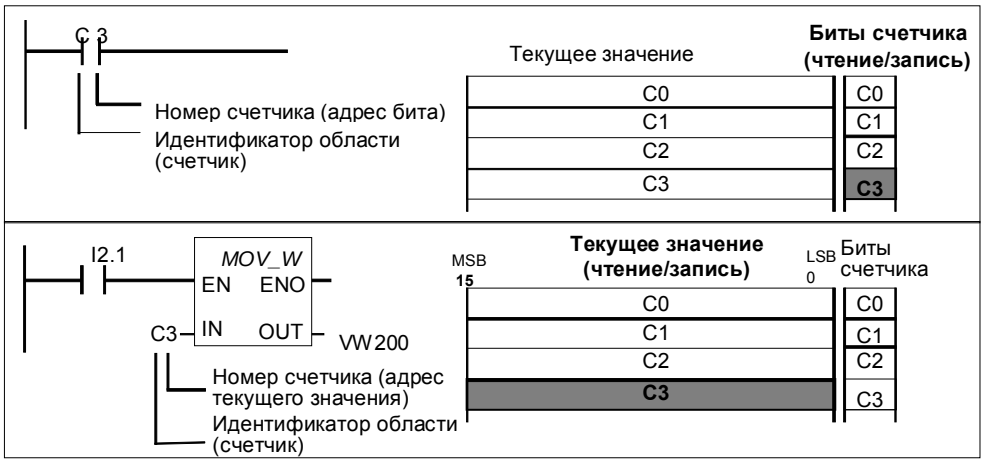

Рис. 5-4. Доступ к данным счетчика SIMATIC

Адресация аналоговых входов (AI)

S7–200 преобразует реальные аналоговые величины (напр., температуру или напряжение) в цифровые значения, имеющее длину, равную слову (16 бит). Доступ к этим величинам производится путем указания идентификатора области (AI), размера данных (W) и начального адреса байта. Так как аналоговые входы являются словами, которые начинаются байтами с четными номерами (напр., 0, 2 или 4), то обращение к этим величинам производится через адреса четных байтов (напр., AIW0, AIW2 или AIW4), как показано на рис. 5–5. Аналоговые входные величины доступны только для чтения.

Формат: **AIW*[начальный адрес байта]*** **AIW4**

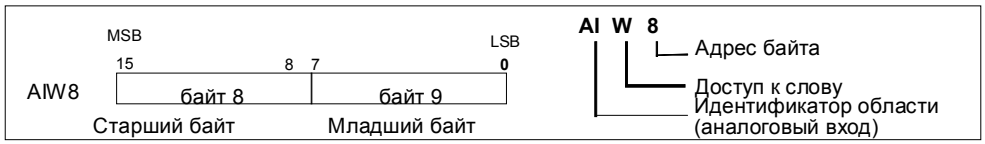

Рис. 5-5. Доступ к аналоговому входу

Адресация аналоговых выходов (AQ)

S7–200 преобразует цифровое значение, имеющее длину, равную слову (16 бит), в ток или напряжение, величина которого пропорциональна цифровому значению. Доступ к этим значениям производится путем указания идентификатора области (AQ), размера данных (W) и начального адреса байта. Так как аналоговые выходы являются словами, которые начинаются байтами с четными номерами (напр., 0, 2 или 4), то они всегда записываются с четными адресами байтов (напр., AQW0, AQW2 или AQW4), как показано на рис. 5–6. Аналоговые выходные величины доступны только для записи.

Формат: **AQW*[начальный адрес байта]***

AQW4

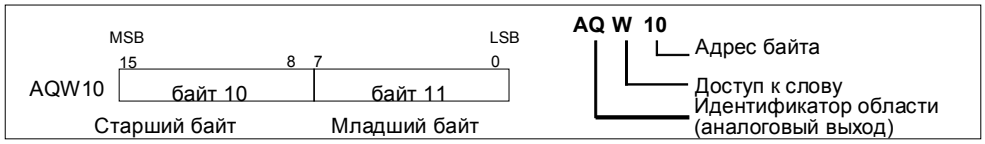

Рис. 5-6. Доступ к аналоговому выходу

Адресация аккумуляторов (AC)

Аккумуляторы – это элементы чтения/записи, которые могут использоваться как память. Например, вы можете использовать аккумуляторы для передачи параметров в подпрограммы и из них и для хранения промежуточных значений, используемых в расчетах. CPU предоставляет в распоряжений четыре 32–битовых аккумулятора (AC0, AC1, AC2 и AC3). Вы можете обращаться к данным в аккумуляторах как к байтам, словам или двойным словам. Как показано на рис. 5–7, для обращения к аккумулятору в формате байтов или слов используются младшие 8 или 16 битов значения, хранящегося в аккумуляторе. Для обращения к аккумулятору в формате двойного слова используются все 32 бита. Размер данных, к которым производится доступ, определяется командой, используемой для обращения к аккумулятору.

Формат: **AC*[номер аккумулятора]*** **AC0**

Примечание
См. раздел «Команды SIMATIC для прерываний и обмена информацией» в главе 9 для получения информации об использовании аккумуляторов с программами прерываний.

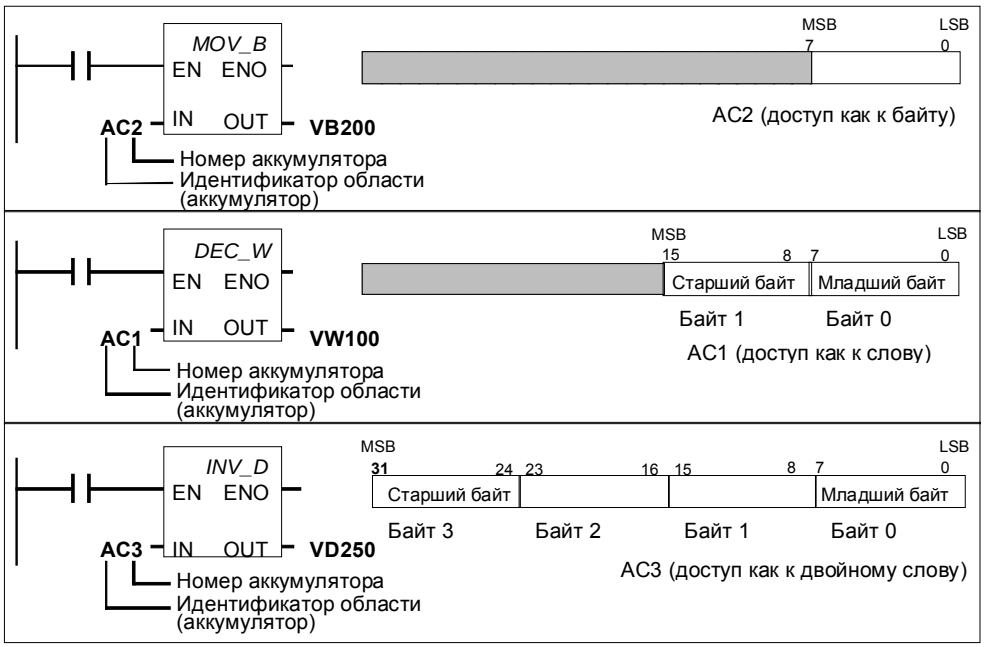

Рис. 5-7. Адресация аккумуляторов

Адресация скоростных счетчиков (HC)

Скоростные счетчики рассчитаны на подсчет событий, происходящих с большой скоростью, независимо от цикла работы CPU. Скоростные счетчики имеют счетное значение (или текущее значение), представляющее собой 32–битовое целое число со знаком. Для доступа к счетному значению скоростного счетчика указывается адрес этого счетчика с помощью типа памяти (HC) и номера счетчика (например, HC0). Текущее значение скоростного счетчика может быть только считано и, как показано на рис. 5–8, может быть адресовано только как двойное слово (32 бита).

Формат: **HC*[номер скоростного счетчика]* HC2**

Рис. 5-8. Доступ к текущему значению скоростного счетчика

Использование констант

Вы можете использовать постоянное значение во многих командах S7–200. Константы могут быть байтами, словами или двойными словами. CPU хранит все константы как двоичные числа, которые затем могут быть представлены в форматах десятичном, шестнадцатеричном, ASCII или с плавающей точкой.

Десятичный формат: **[десятичное значение]**
Шестнадцатеричный формат: **16#[шестнадцатеричное значение]**
Формат ASCII: **'[текст ASCII]'**
Формат с плавающей точкой (или вещественный): **ANSI/IEEE 754-1985**

Двоичный формат в виде: **2#1010_0101_1010_0101**

CPU S7–200 не поддерживает и не проверяет типы данных (например, указание на то, что константа хранится как целое, целое со знаком или двойное целое число). Например, команда «Сложить» может использовать значение в VW100 как целое число со знаком, а команда «Исключающее ИЛИ» может использовать то же самое значение в VW100 как двоичное число без знака.

Следующие примеры показывают константы для форматов: десятичного, шестнадцатеричного, ASCII и с плавающей точкой:

- Десятичная константа: **20047**
- Шестнадцатеричная константа: **16#4E4F**
- Константа ASCII: **'Текст в одиночных кавычках.'**
- Формат с плавающей точкой (или вещественный):
  +1.175495E-38 (положительное число)
  -1.175495E-38 (отрицательное число)
- Двоичный формат **2#1010_0101_1010_0101**

5.2 Косвенная адресация областей памяти CPU в SIMATIC

Косвенная адресация использует указатель для доступа к данным в памяти. CPU 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 аналоговым значениям.

Создание указателя

Для косвенного обращения к адресу в памяти вы сначала должны создать указатель на этот адрес. Указатели – это ячейки памяти, имеющие размер двойного слова, которые содержат адрес другой ячейки памяти. В качестве указателей можно использовать только ячейки V-памяти, L-памяти или аккумуляторные регистры (AC1, AC2, AC3). Для создания указателя вы должны использовать команду «Переместить двойное слово» (MOVD), чтобы переместить адрес косвенно адресованной ячейки памяти в ячейку указателя. Входному операнду команды должен предшествовать амперсанд (&), чтобы указать на необходимость перемещения в ячейку, обозначенную в выходном операнде команды (указатель), адреса ячейки памяти, а не ее содержимого.

Пример:	***MOVD***	***&VB100, VD204***
	MOVD	***&MB4, AC2***
	MOVD	***&C4, LD6***

Использование указателя для доступа к данным

Ввод астериска (*) перед операндом команды указывает, что этот операнд является указателем. В примере, показанном на рис. 5–9, *AC1 указывает, что AC1 является указателем на слово, на которое ссылается команда «Переместить слово» (MOVW). В этом примере значения, хранящиеся в V200 и V201, перемещаются в аккумулятор AC0.

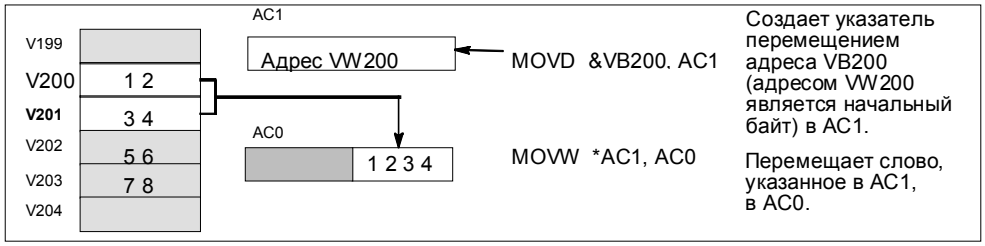

Рис. 5-9. Использование указателя для косвенной адресации

Изменение указателей

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. Не забывайте указывать размер данных, к которым вы обращаетесь:

- При обращении к байтам увеличивайте значение указателя на единицу.
- При обращении к слову или текущему значению таймера или счетчика увеличивайте значение указателя на два.
- При обращении к двойному слову увеличивайте значение указателя на четыре.

Рис. 5–10 дает пример того, как создать указатель для косвенной адресации, как обращаться к данным косвенно и как можно увеличивать указатель.

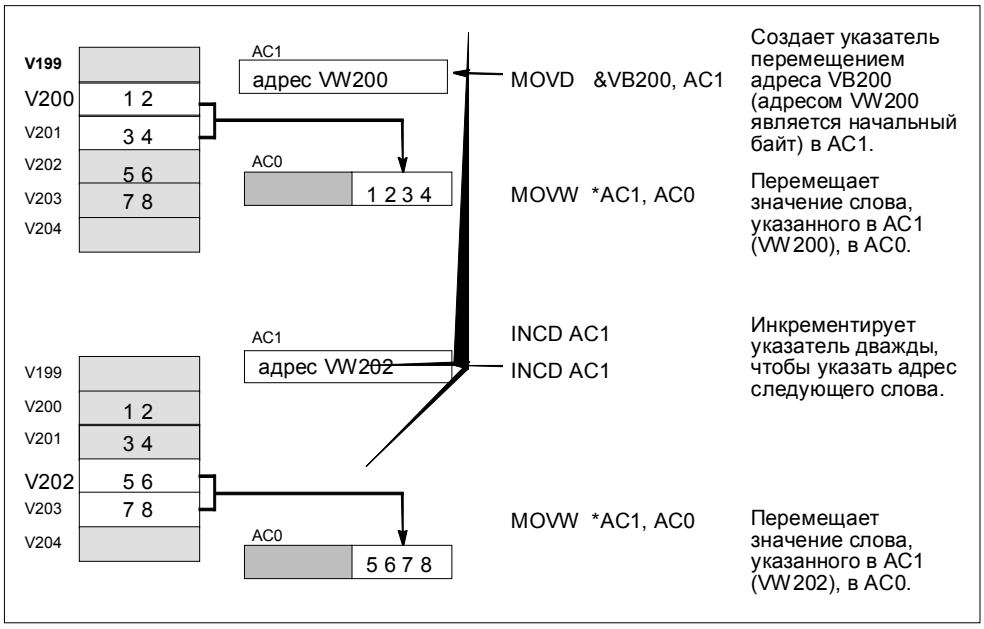

Рис. 5-10. Изменение указателя при обращении к слову

5.3 Хранение данных в CPU S7–200

CPU S7–200 предоставляет несколько методов, гарантирующих, что ваша программа, данные программы и конфигурационные данные вашего CPU будут сохраняться надлежащим образом. См. рис. 5–11.

- CPU предоставляет в распоряжение ЭСППЗУ для постоянного, устойчивого к исчезновению напряжения, хранения всей вашей программы, выбранных пользователем областей данных и конфигурационных данных вашего CPU.
- CPU предоставляет в распоряжение конденсатор большой емкости, который поддерживает целостность ОЗУ после отключения питания CPU. В зависимости от CPU этот конденсатор может поддерживать ОЗУ в течение нескольких дней.
- CPU 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.

В этом разделе обсуждаются вопросы постоянного хранение и сохранения данных в ОЗУ при различных обстоятельствах.

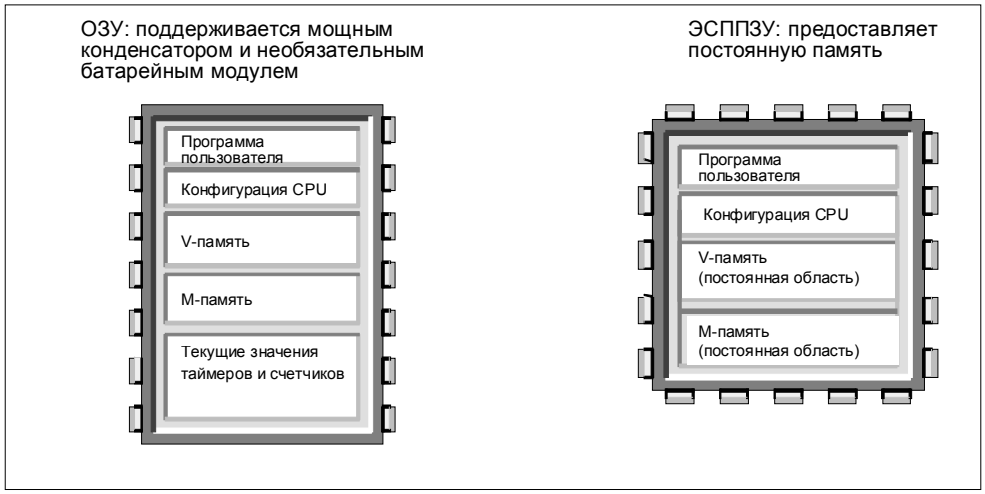

Рис. 5-11. Области памяти CPU S7-200

Загрузка вашего проекта в CPU и из CPU

Ваш проект состоит из трех элементов: программы пользователя, блока данных (не обязателен) и конфигурации CPU (не обязательна). Как показано на рис. 5–12, загрузка проекта в CPU сохраняет эти элементы в ОЗУ CPU. CPU также автоматически копирует программу пользователя, блок данных (DB1) и конфигурацию CPU в ЭСППЗУ для постоянного хранения.

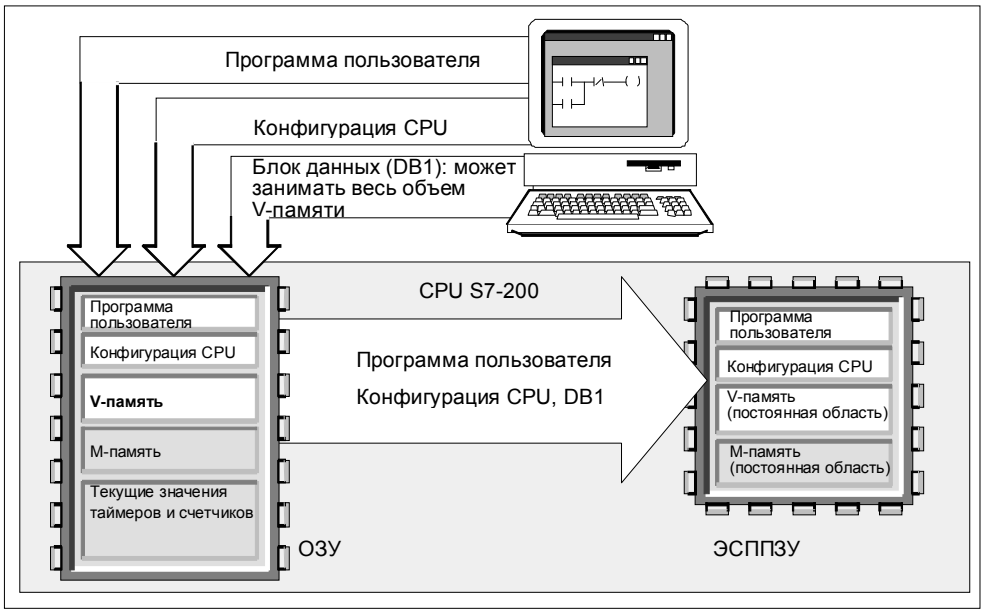

Рис. 5-12. Загрузка элементов проекта в CPU

Когда вы загружаете проект из CPU, как показано на рис. 5–13, конфигурация CPU загружается из ОЗУ в ваш компьютер. Программа пользователя и постоянная область V-памяти загружаются в ваш компьютер из ЭСППЗУ.

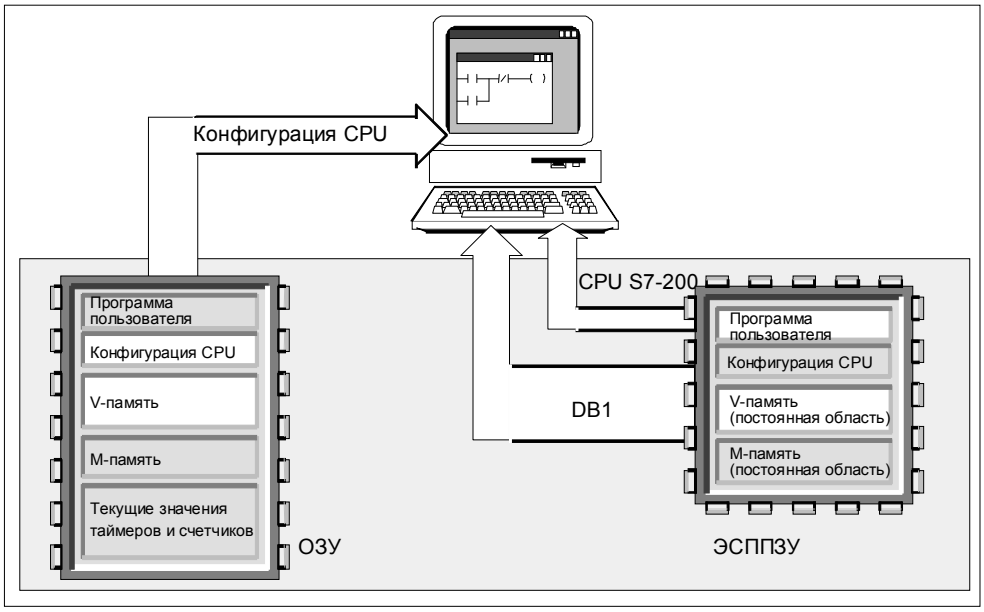

Рис. 5-13. Загрузка элементов проекта из CPU

Автоматическое сохранение данных из битовой памяти (M) при потере питания CPU

Если первые 14 байтов M-памяти (от MB0 до MB13) были определены при конфигурировании как реманентные (сохраняемые), то они сохраняются в ЭСППЗУ, когда CPU теряет питание. Как показано на рис. 5–14, CPU передает эти сохраняемые области из M-памяти в ЭСППЗУ. В STEP 7–Micro/WIN 32 по умолчанию эта возможность отключена (off).

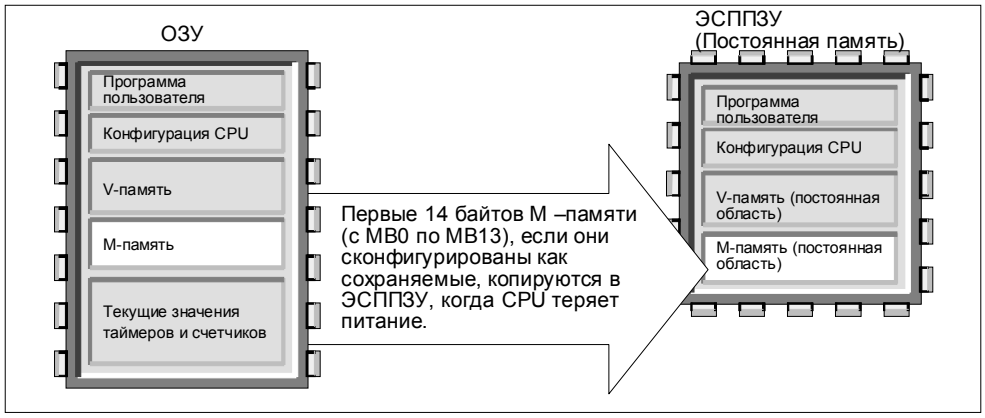

Рис. 5-14. Сохранение части битовой памяти (M) в ЭСППЗУ при потере питания

Восстановление памяти при включении питания

При включении питания CPU восстанавливает программу пользователя и конфигурацию CPU, извлекая их из ЭСППЗУ. См. рис. 5–15.

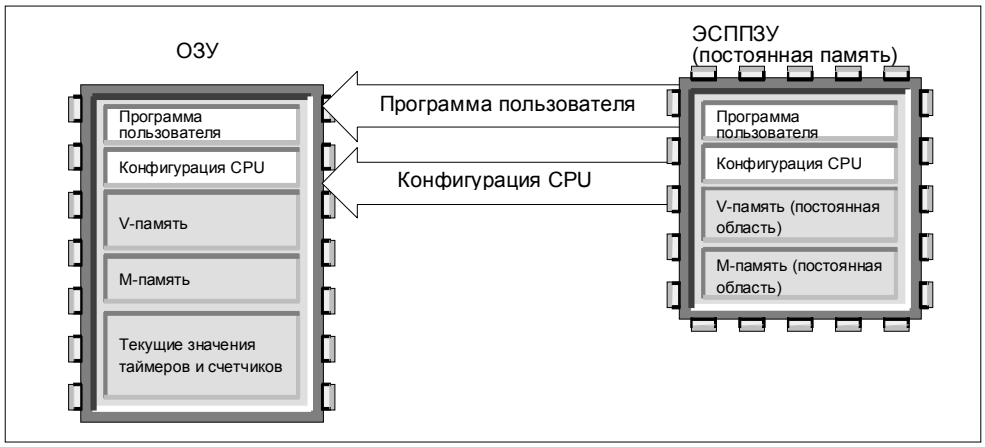

Рис. 5-15. Восстановление программы пользователя и конфигурации CPU при включении питания

При включении питания CPU 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 реманентные области ОЗУ остаются неизменными. Как показано на рис.
5–16, неременантные области V-памяти восстанавливаются из соответствующей постоянной области V-памяти в ЭСППЗУ.

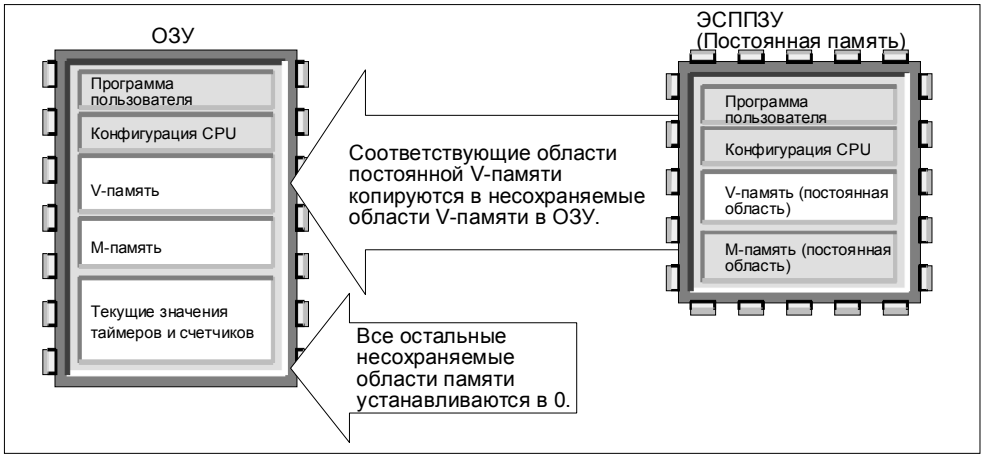

Рис. 5-16. Восстановление данных программы при включении питания (данные были успешно сохранены в ОЗУ)

Если содержимое ОЗУ не было сохранено (например, после длительного перерыва в питании), CPU очищает ОЗУ (включая сохраняемую и несохраняемую области) и устанавливает бит памяти Retentive Data Lost [Сохраняемые данные потеряны] (SM0.2) для первого цикла обработки программы, следующего за включением питания. Как показано на рис. 5–17, данные, сохраненные в постоянной памяти ЭСППЗУ, затем копируются в ОЗУ.

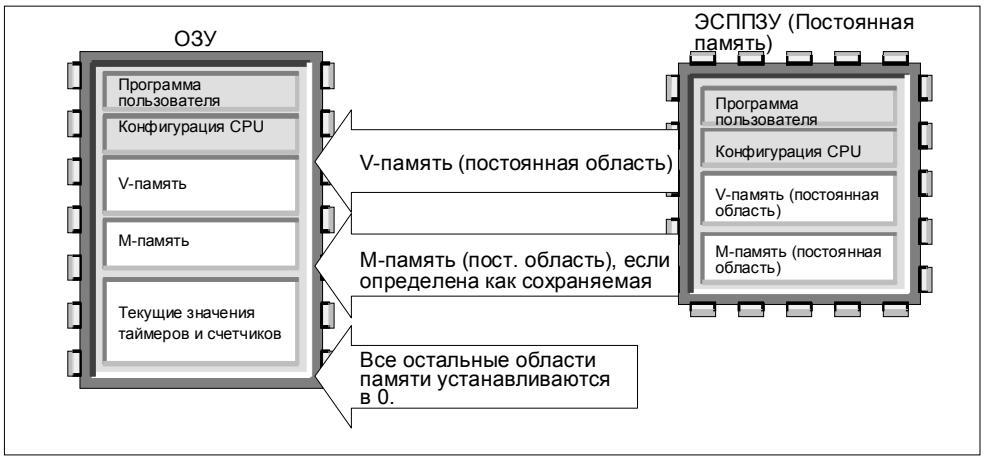

Рис. 5-17. Восстановление данных программы при включении питания (данные в ОЗУ не сохранены)

Определение сохраняемых областей памяти

Вы можете определить до шести сохраняемых областей и выбрать области памяти, которые вы хотите буферизовать при потере питания (см. рис. 5–18). Вы можете определить в качестве сохраняемых диапазоны адресов в следующих областях памяти: V, M, C и T. Из таймеров могут быть буферизованы только реманентные (сохраняемые) таймеры (TONR). В STEP 7-Micro/WIN 32 по умолчанию первые 14 байтов M-памяти определены как несохраняемые. Это умолчание блокирует сохранение, которое нормально происходит, когда вы выключаете CPU.

Примечание
Для таймеров и счетчиков могут быть сохранены только текущие значения: биты таймеров и счетчиков нереманентны.

Для определения сохраняемых диапазонов для областей памяти выберите команду меню **View** → **System Block** [Вид → Системный блок] и щелкните на закладке Retentive Ranges [Сохраняемые диапазоны]. Диалоговое окно для определения конкретных диапазонов, которые должны быть сохраняемыми, показано на рис. 5–18. Для получения сохраняемых диапазонов, принятых по умолчанию для вашего CPU, нажмите кнопку **Defaults [Умолчания]**.

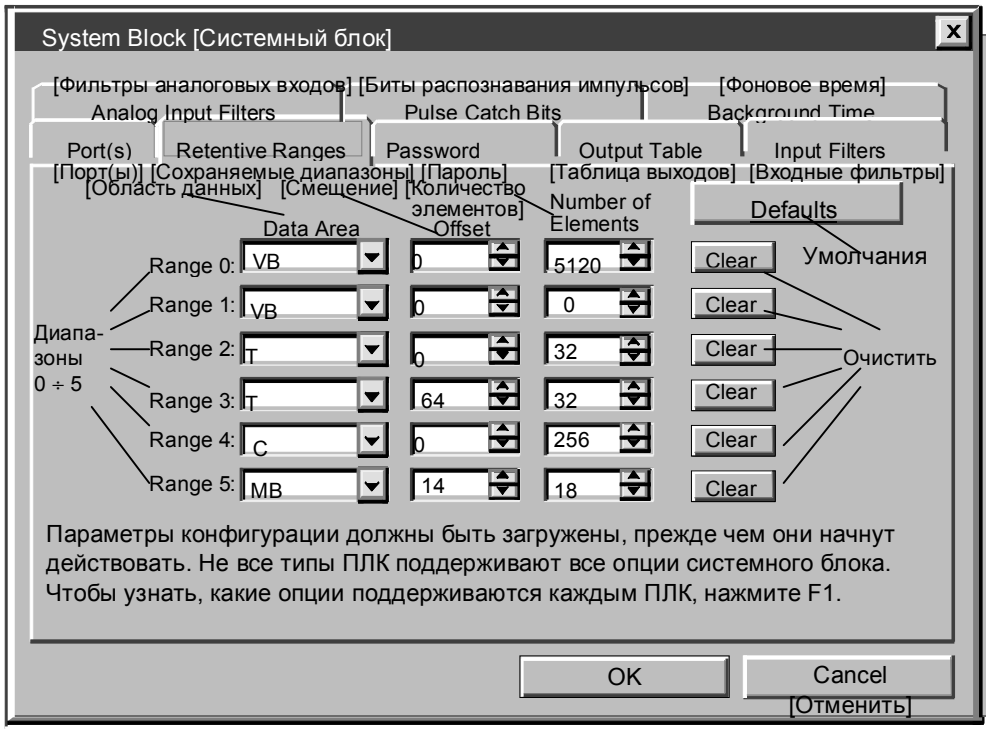

Рис. 5-18. Конфигурирование сохраняемых диапазонов для памяти CPU

5.4 Сохранение данных в постоянной памяти с помощью вашей программы

Вы можете сохранить значение (байт, слово или двойное слово), находящееся в V-памяти, в ЭСППЗУ. Это свойство может быть использовано для сохранения значения в любой ячейке постоянной области V-памяти.

Операция сохранения в ЭСППЗУ обычно удлиняет время цикла обработки программы до 5 мс. Значение, записанное операцией сохранения заменяет любое предыдущее значение, хранящееся в постоянной области V-памяти ЭСППЗУ.

Примечание
Операция сохранения в ЭСППЗУ не обновляет данные в модуле памяти.

Копирование V-памяти в ЭСППЗУ

Байт 31 специальной памяти (SMB31) и слово 32 специальной памяти (SMW32) дают CPU команду скопировать значение из V-памяти в постоянную область V-памяти ЭСППЗУ. Рис. 5–19 показывает формат SMB31 и SMW32. Выполните следующие шаги, чтобы запрограммировать CPU на сохранение или запись конкретного значения в V-памяти:

1. Загрузите адрес значения в V-памяти, которое вы хотите сохранить, в SMW32.
2. Загрузите размер данных в SM31.0 и SM31.1. (См. рис. 5–19.)
3. Установите SM31.7 в 1.

В конце каждого цикла выполнения программы CPU проверяет SM31.7; если SM31.7 равен 1, то указанное значение сохраняется в ЭСППЗУ. Операция завершается, когда CPU сбрасывает SM31.7 в 0. Не изменяйте значение в V-памяти, пока операция сохранения не будет завершена.

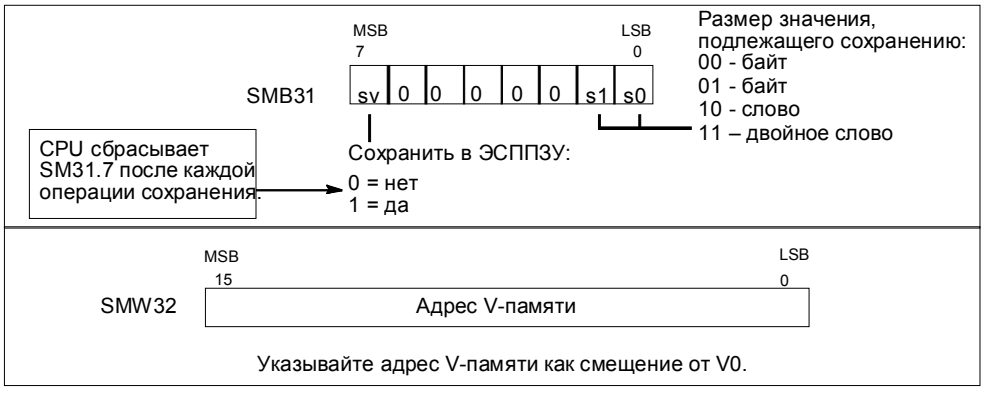

Рис. 5-19. Формат SMB31 и SMW32

Ограничение числа программируемых операций сохранения в ЭСППЗУ

Так как число операций сохранения в ЭСППЗУ ограничено (100 000 минимум, 1 000 000 типично), 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.

5.5 Использование модуля памяти для хранения вашей программы

CPU поддерживают необязательный модуль памяти, который предоставляет в распоряжение вашей программе сменное ЭСППЗУ. CPU хранит в модуле памяти следующие элементы:

- программу пользователя
- данные, хранимые в постоянной области V-памяти ЭСППЗУ
- конфигурацию CPU (системный блок)

Дополнительную информацию о модуле памяти вы найдете в приложении A.

Копирование в модуль памяти

Вы можете скопировать свою программу в модуль памяти из ОЗУ только тогда, когда CPU получает питание и находится в состоянии STOP, а модуль памяти вставлен.

Осторожно
Электростатический разряд может повредить модуль памяти или гнездо на CPU.
При работе с модулем вы должны находиться на заземленном проводящем полу и/или носить на руке заземленный браслет. Храните модуль в токопроводящем контейнере.

Модуль памяти можно вставлять и вынимать, когда CPU включен. Для установки модуля памяти снимите с ПЛК пластмассовую крышку и вставьте модуль памяти в ПЛК. (Модуль памяти имеет такую форму, что он может быть вставлен только должным образом.) После того как модуль памяти вставлен, используйте для копирования программы следующую процедуру:

1. Переведите CPU в состояние STOP.
2. Загрузите программу в CPU, если она еще не была загружена.
3. Используйте команду меню **PLC** → **Program Memory Cartridge [ПЛК** → **Запрограммировать модуль памяти]**, чтобы скопировать программу в модуль памяти. Рис. 5–20 показывает элементы памяти CPU, которые сохраняются в модуле памяти.
4. Вытащите модуль памяти из гнезда (не обязательно).

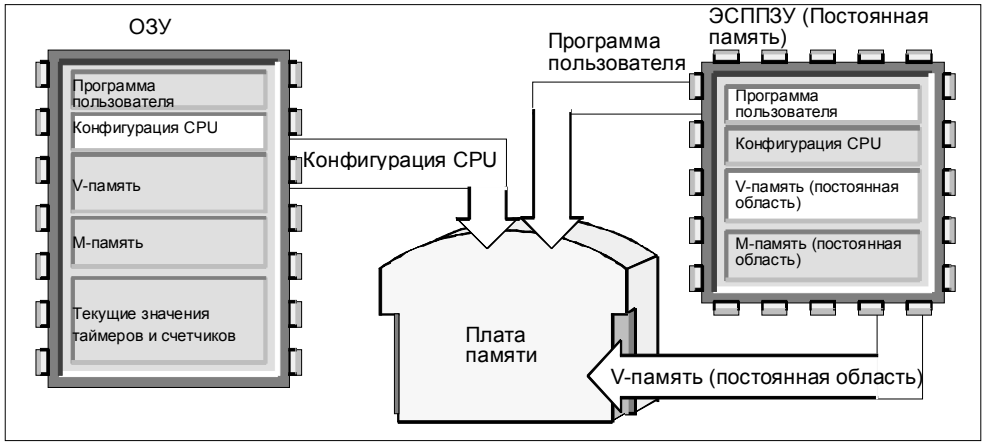

Рис. 5-20. Копирование памяти CPU в модуль памяти

Восстановление программы и памяти из модуля памяти

Для передачи программы из модуля памяти в CPU вы должны выключить, а затем снова включить питание CPU при вставленном модуле памяти. Как показано на рис. 5–21, CPU после включения питания выполняет следующие задачи (при вставленном модуле памяти):

- ОЗУ очищается.
- Содержимое модуля памяти копируется в ОЗУ.
- Программа пользователя, конфигурация CPU и область V-памяти копируются в постоянное ЭСППЗУ.

Примечание
Включение CPU с пустым модулем памяти или с модулем памяти, запрограммированным в другой модели CPU, 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. После этого модуль памяти может быть снова вставлен и перепрограммирован, если необходимо.

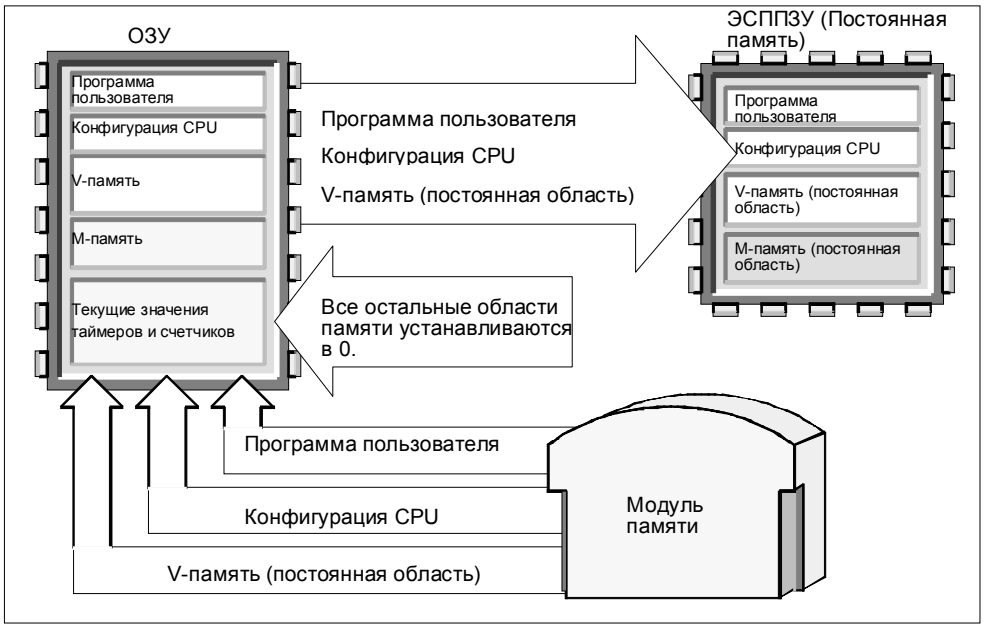

Рис. 5-21. Восстановление памяти при включении питания (при установленном модуле памяти)