Память CPU: Типы данных и способы адресации

6

CPU S7–200 имеет в своем распоряжении специфические модули памяти для того, чтобы Ваши данные могли обрабатываться быстрее и значительно более эффективно.

Обзор главы

6.1 Прямая адресация областей памяти в CPU

CPU S7–200 хранит информацию в различных ячейках памяти, которые имеют уникальные адреса. Адреса памяти, к которым Вы хотите обратиться, можно указывать в явном виде. Благодаря этому Ваша программа имеет прямой доступ к информации.

Обращение к данным через адреса

Если Вы хотите обратиться к биту в области памяти, то Вы должны указать адрес бита. Этот адрес состоит из идентификатора области памяти, адреса байта, а также номера бита □такая адресация называется также адресацией “байт.бит”). В данном примере за идентификатором области памяти и адресом байта □E = вход, 3 = байт 3) следует точка □”.”), чтобы отделить адрес бита □бит 4).

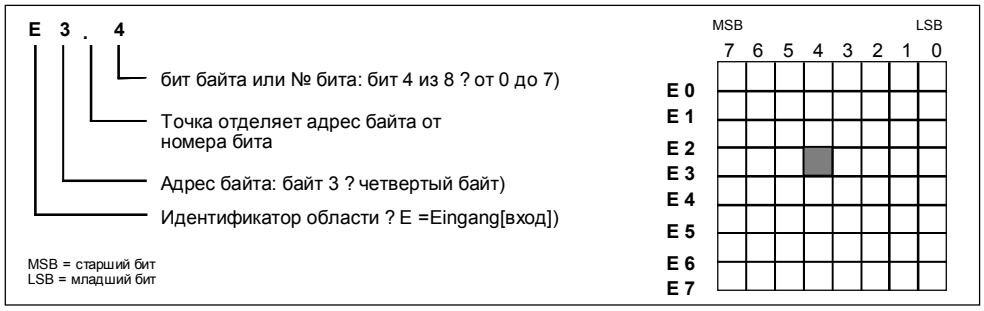

Рис. 6-1. Обращение к биту данных в памяти CPU (адресация байт.бит)

Когда Вы используете для адресации формат байта, Вы можете обращаться к данным в различных областях памяти CPU (V, E, A, M и SM) как к байтам, словам или двойным словам. Если Вы хотите обратиться к байту, слову или двойному слову, то Вы должны задать этот адрес наподобие адреса бита. Вы указываете идентификатор области, размер данных (формат доступа) и начальный адрес значения в формате байта, слова или двойного слову □см. рис. 6–2). Обращение к данным в других областях памяти CPU □например, T, Z, HC и аккумуляторы) производится указанием в качестве адреса идентификатора области и номера элемента.

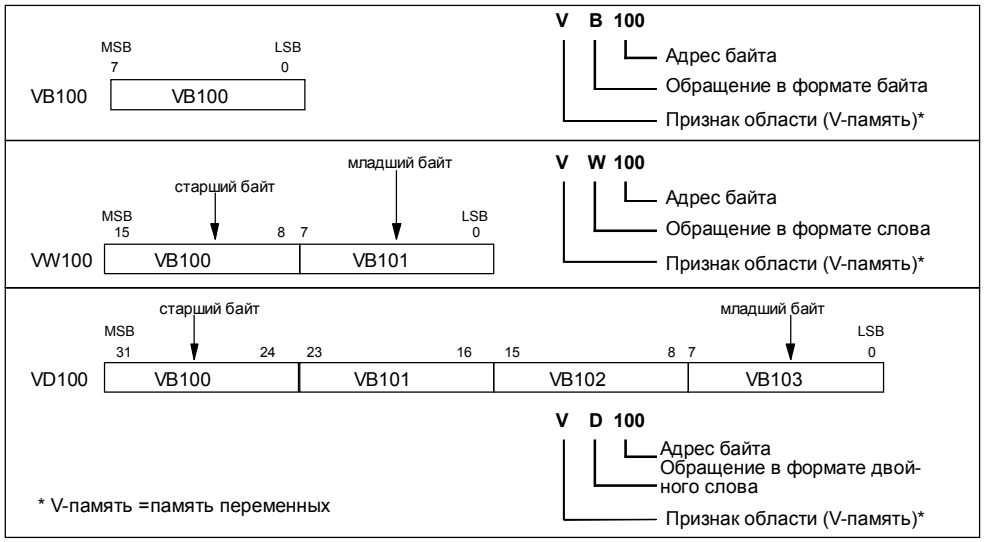

Рис. 6-2. Обращение к одному и тому же адресу в формате байта, слова и двойного слова

Представление чисел

В табл. 6–1 показана область целочисленных значений, которые могут представляться данными различного размера.

Вещественные числа (числа с плавающей точкой) представляются как числа одинарной точности 32 бита), формат которых описан в стандарте ANSI/IEEE 754–1985. Обращение к значениям вещественных чисел производится в формате двойного слова.

Таблица 6–1. Обозначение размера данных (и соответствующие диапазоны целых чисел)

Размер данных	Диапазон целых чисел без знака		Диапазон целых чисел со знаком	
	Десятичный	Шестнадца-тиричный	Десятичный	Шестнадцатиричный
B (байт): 8 битов	от 0 до 255	от 0 до FF	от -128 до 127	от 80 до 7F
W (слово): 16 битов	от 0 до 65.535	от 0 до FFFF	от -32.768 до 32.767	от 8000 до 7FFF
D (двойное слово): 32 бита	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 области отображения процесса на входах (E)

Как описано в разделе 5.5, в начале каждого цикла CPU опрашивает физические входы и записывает эти значения в область отображения процесса на входах. Вы можете обращаться к этой области отображения процесса в формате бита, байта, слова или двойного слова.

Формат: бит **E*[адрес байта].[адрес бита]*** **E0.1**
байт, слово, двойное слово
E*[размер][начальный адрес байта]* **EB4**

Адресация области отображения процесса на выходах A)

В конце цикла CPU копирует значения из области отображения процесса на выходах на физические выходы. Вы можете обращаться к этой области отображения процесса в формате бита, байта, слова или двойного слова.

Формат: бит **A*[адрес байта].[адрес бита]*** **A1.1**
байт, слово, двойное слово
A*[размер][начальный адрес байта]* **AB5**

Адресация памяти переменных (V)

В памяти переменных можно хранить промежуточные результаты, рассчитываемые операциями Вашей программы. Вы можете хранить в памяти переменных также другие данные, которые относятся к Вашему процессу или к Вашему решению задачи автоматизации. К памяти переменных можно обращаться в формате бита, байта, слова или двойного слова.

Формат: бит **V*[адрес байта].[адрес бита]*** **V10.2** байт, слово, двойное слово
V*[размер][начальный адрес байта]* **VW100**

Адресация меркеров (M)

Внутренние меркеры область памяти меркеров, M) можно использовать как управляющие реле для того, чтобы сохранять промежуточные результаты операций или другую управляющую информацию. Вы можете обращаться к меркерам в формате бита, байта, слова или двойного слова.

Формат: бит **M*[адрес байта].[адрес бита]*** **M26.7**
байт, слово, двойное слово
M*[размер][начальный адрес байта]* **MD20**

Адресация реле шагового управления (S)

С помощью реле шагового управления (S) Вы расчленяете алгоритм функционирования установки на отдельные шаги или эквивалентные программные компоненты. С помощью реле шагового управления Вы можете логически структуризовать Вашу управляющую программу. Вы можете обращаться к S–битам в формате бита, байта, слова или двойного слова.

Формат: бит **S*[адрес байта].[адрес бита]*** **S3.1**
байт, слово, двойное слово
S*[размер][начальный адрес байта]* **SB4**

Адресация специальных меркеров (SM)

С помощью специальных меркеров можно производить обмен информацией между CPU и Вашей программой. Кроме того, специальные меркеры служат для того, чтобы выбирать особые функции CPU S7–200 и управлять ими. К ним относятся:

- Бит, который включается только в первом цикле.

- Биты, которые включаются и выключаются на определенных тактах.
- Биты, которые отображают состояние арифметических и других операций.

Подробную информацию о специальных меркерах Вы найдете в приложении D. Область памяти специальных меркеров основывается на битах, однако Вы можете обращаться к данным в этой области в формате бита, байта, слова или двойного слова.

Формат: бит **SM*[адрес байта].[адрес бита]*** **SM0.1**
байт, слово, двойное слово
SM*[размер][начальный адрес байта]* **SMB86**

Адресация таймеров (T)

В CPU S7–200 таймеры являются элементами, подсчитывающими приращения времени. Таймеры S7–200 имеют разрешающую способность (приращения, определяемые базой времени) 1 мс, 10 мс и 100 мс. Каждый таймер имеет в своем распоряжении следующие две переменные:

- Текущее значение: Это целое число (16 битов) со знаком хранит значение времени таймера.
- Бит таймера: Этот бит включается (устанавливается в “1”), когда текущее значение таймера больше или равно предварительно установленному значению (предварительно устанавливаемое значение вводится вместе с операцией).

Обращение к обоим элементам данных производится через адрес таймера (T + номер таймера). Куда происходит обращение - к биту таймера или к текущему значению таймера - определяется по соответствующей операции. Операции с операндами в формате бита обеспечивают доступ к биту таймера, тогда как операции с операндами в формате слова обеспечивают доступ к текущему значению. На рисунке 6–3 Вы видите, что операция “замыкающий контакт” обеспечивает доступ к биту таймера, тогда как операция передачи слова (MOVW) обеспечивает доступ к текущему значению таймера. Подробную информацию об операциях S7–200 Вы найдете в главе 9.

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

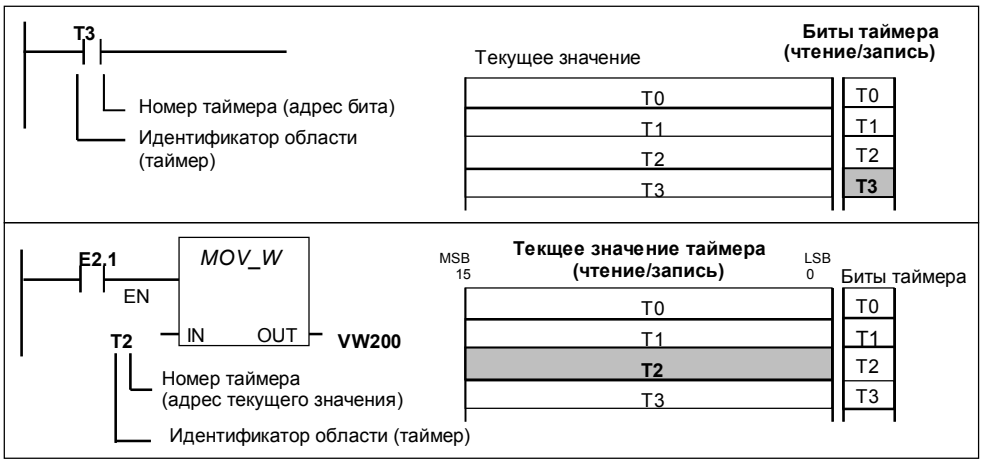

Рис. 6-3. Обращение к данным таймера

Адресация счетчиков (Z)

В CPU S7–200 счетчики являются элементами, подсчитывающими нарастающие фронты сигнала на счетных входах. CPU имеет в своем распоряжении счетчики двух видов: счетчик первого вида считает только вперед, тогда как счетчик другого вида считает как вперед, так и назад. Каждый счетчик имеет в своем распоряжении следующие две переменные:

- Текущее значение: Это целое число (16 битов) со знаком хранит накопленное значение счетчика.
- Бит счетчика: Этот бит включается (устанавливается в “1”), когда текущее значение счетчика больше или равно предварительно установленному значению (предварительно устанавливаемое значение вводится вместе с операцией).

Доступ к обеим переменным осуществляется через адрес счетчика (Z + номер счетчика). Куда происходит доступ - к биту счетчика или к текущему значению счетчика - определяется по соответствующей операции. Операции с операндами в формате бита обеспечивают доступ к биту счетчика, тогда как операции с операндами в формате слова обеспечивают доступ к текущему значению. На рисунке 6–4 Вы видите, что операция “замыкающий контакт” обеспечивает доступ к биту счетчика, тогда как операция передачи слова (MOVW) обеспечивает доступ к текущему значению счетчика. Подробную информацию об операциях S7–200 Вы найдете в главе 9.

Формат: ***Z[номер счетчика]*** **Z20**

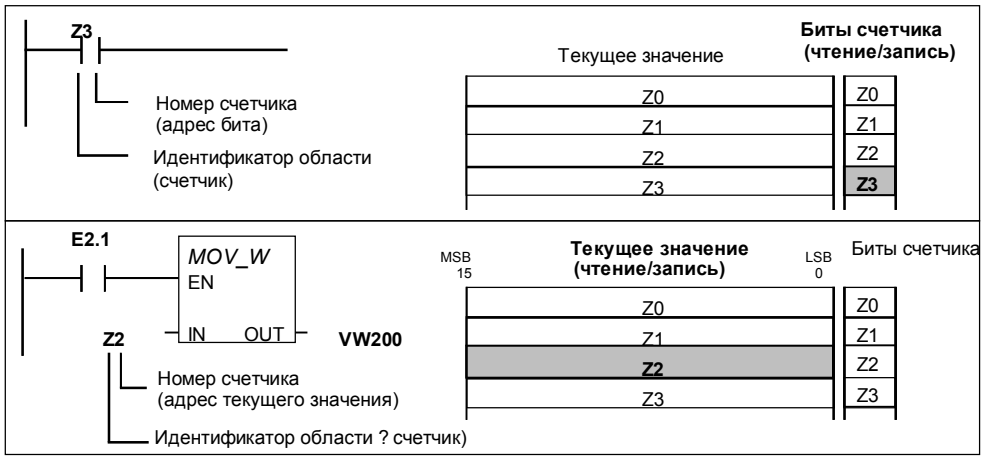

Рис. 6-4. Обращение к данным счетчика

Адресация аналоговых входов (AE)

S7–200 преобразует реальные аналоговые значения (например, напряжение, температуру) в цифровые значения с разрядностью слова (16 битов). Доступ к этим значениям осуществляется через идентификатор области (AE), размер данных (W) и начальный адрес байта. Так как в случае аналоговых входов речь идет о словах, всегда начинающихся с четного байта (следовательно, с байта 0, 2, 4 и т.д.), то обращение к ним производится с использованием адресов четных байтов (например, AEW0, AEW2, AEW4) (см. рис. 6–5). Аналоговые входы могут только считываться.

Формат: **AEW*[начальный адрес байта]*** **AEW4**

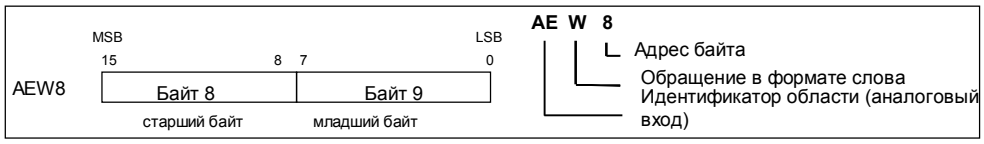

Рис. 6-5. Обращение к аналоговому входу

Адресация аналоговых выходов (AA)

S7–200 преобразует цифровые значения с разрядностью слова (16 битов) в ток или напряжение пропорционально цифровому значению. Обращение к этим значениям производится через идентификатор области (AA), размер данных (W) и начальный адрес байта. Так как в случае аналоговых выходов речь идет о словах, всегда начинающихся с четного байта (следовательно, с байта 0, 2, 4 и т.д.), то обращение к ним производится с использованием адресов четных байтов (например, AAW0, AAW2, AAW4) (см. рис. 6–6). Ваша программа не может считывать значения аналоговых выходов.

Формат: **AAW*[начальный адрес байта]*** **AAW4**

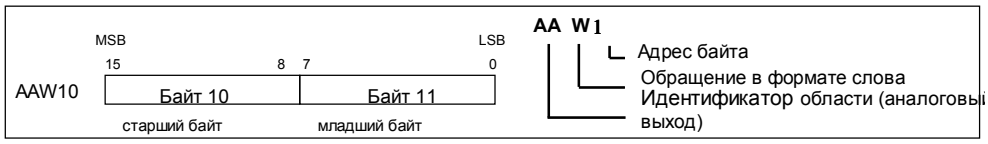

Рис. 6-6. Обращение к аналоговому выходу

Адресация аккумуляторов

Аккумуляторы являются элементами для чтения/записи, которые используются как память. С помощью аккумуляторов Вы можете, например, передавать параметры в подпрограмму, а также принимать их обратно, или сохранять промежуточные результаты вычислений. CPU имеет в своем распоряжении четыре 32-битных аккумулятора (AC0, AC1, AC2 и AC3). Обращение к данным в аккумуляторах может производиться в формате бита, байта, слова и двойного слова. Если обращение к аккумулятору производится в формате байта или слова, то при этом используются младшие 8 или 16 битов значения, хранимого в аккумуляторе. Если обращение к аккумулятору производится в формате двойного слова, то используются все 32 бита. Размер данных, к которым производится доступ, определяется операцией, посредством которой Вы обращаетесь к аккумулятору.

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

Указание
Информацию об использовании аккумуляторов программами обработки прерываний Вы найдете в разделе 9.15.

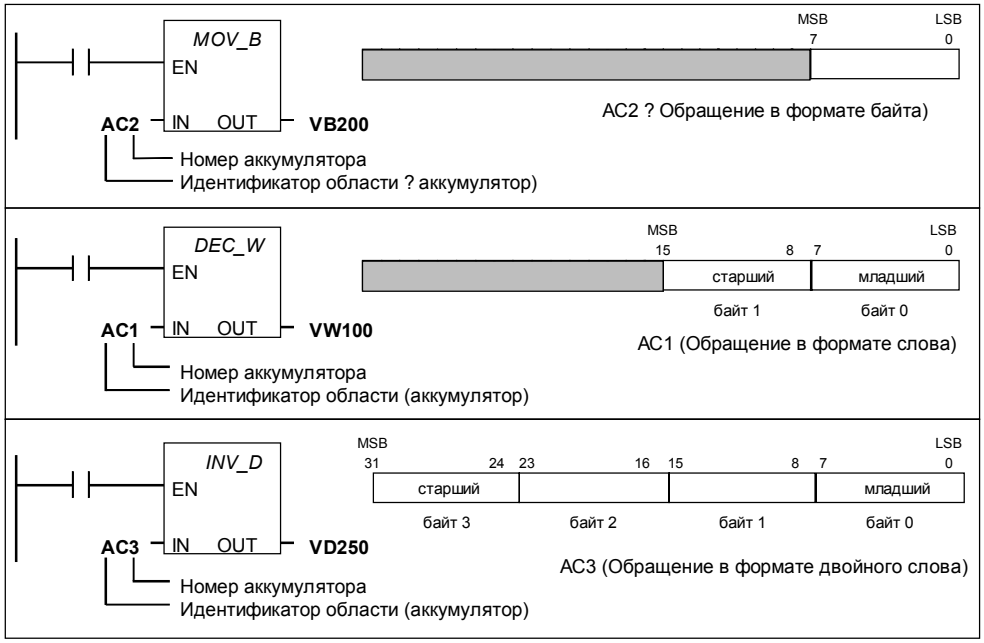

Рис. 6-7. Адресация аккумуляторов

Адресация быстрых счетчиков

Быстрые счетчики подсчитывают события быстрее, чем CPU может опрашивать эти события. Быстрые счетчики имеют в своем распоряжении 32-битное счетное значение (текущее значение). Если Вы хотите обратиться к счетному значению быстрого счетчика, то задайте адрес быстрого счетчика посредством идентификатора области (HC) и номера счетчика (например, HC0). Текущее значение счетчика защищено от записи и может адресоваться только в формате двойного слова (32 бита)
(см. рис. 6–8).

Формат: **HC*[номер счетчика]*** **HC1**

Рис. 6-8. Обращение к текущему значению быстрого счетчика

Применение констант

Во многих операциях для S7–200 Вы можете использовать константы. Константы могут быть байтами, словами и двойными словами. CPU хранит все константы как двоичные значения, которые могут представляться в десятичном, шестнадцатиричном и ASCII–формате.

Десятичный формат:	**[десятичное значение]**
Шестнадцатиричный формат:	**16#[шестнадцатиричное значение]**
ASCII–формат:	**'[ASCII–текст]'**

Вы не можете задавать в CPU S7–200 специфические типы данных (например, когда Вы хотите указать, что константа должна храниться в виде целого числа (16 битов), целого числа со знаком или целого числа (32 бита)). CPU S7–200 также не проверяет типы данных. Например, операция сложения может использовать хранимое в VW100 значение как целое число со знаком, тогда как операция EXKLUSIV ODER использует то же самое значение из VW100 как двоичное значение без знака.

Следующие примеры показывают константы в десятичном, шестнадцатиричном и ASCII–формате:

- Десятичная константа: **20047**
- Шестнадцатиричная константа: **16#4E4F**
- ASCII–константа: **'Текст в апострофах.'**

6.2 Косвенная адресация областей памяти в CPU

Косвенная адресация использует указатели для обращения к данным в памяти. В CPU S7–200 посредством указателей можно косвенно адресовать следующие области памяти: E, A, V, M, S, T (только текущее значение) и Z (только текущее значение). Нельзя косвенно адресовать значения отдельных битов.

Создание указателя

Если Вы хотите обратиться к адресу косвенно, то Вы должны вначале создать указатель, указывающий на этот адрес. Указатели являются двойными словами. Для создания указателя используется операция передачи двойного слова (MOVD). Эта операция передает адрес в ячейку памяти с другим адресом или в аккумулятор, которая или который, соответственно, служит потом указателем (см. рис. 6–9). С помощью знака ”&” (амперсанд) указывается, что именно адрес, а не соответствующее ему значение должно передаваться в пункт назначения.

Формат: ***&[адрес памяти]*** **&MB16**

При создании указателя Вы можете задавать в операции MOVD в качестве целевого адреса только адреса памяти переменных и аккумуляторы AC1, AC2 и AC3. При косвенной адресации нельзя использовать в качестве указателя AC0.

Доступ к данным с помощью указателя

Звездочка (*) перед операндом операции указывает, что этот операнд является указателем. В примере на рисунке 6–9 запись *AC1 указывает, что AC1 является указателем, который содержит адрес значения слова, заданного операцией “передача слова” (MOVW). В данном примере значения V200 и V201 передаются в аккумулятор AC0.

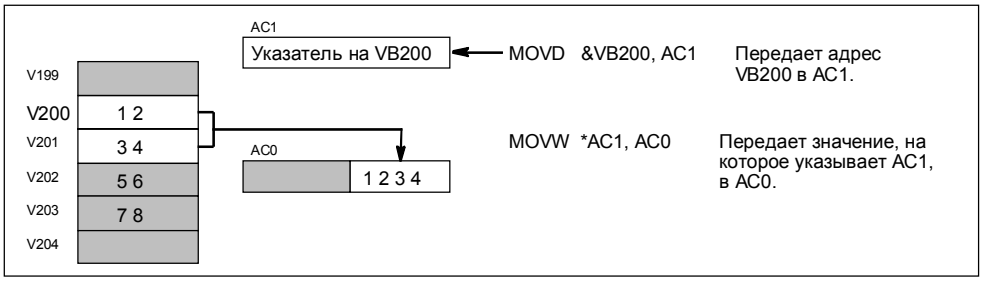

Рис. 6-9. Применение указателя для косвенной адресации

Изменение указателей

Вы можете изменять значение указателя. Так как указатели являются 32-битными значениями, то Вы должны изменять значения указателей с помощью операций для двойных слов. Изменять значения указателей можно с помощью простых арифметических операций, например, путем сложения или инкрементирования. Обратите внимание на размер данных, к которым Вы хотите обратиться:

- Если Вы обращаетесь к байту, увеличьте значение указателя на 1.
- Если Вы обращаетесь к словам или к текущим значениям таймеров или счетчиков, то прибавляйте значение 2 или инкрементируйте значение указателя на 2.
- Если Вы обращаетесь к двойному слову, то прибавляйте значение 4 или инкрементируйте значение указателя на 4.

6.3 Сохранение данных в CPU S7–200

CPU S7–200 предоставляет различные методы для надежного хранения Вашей программы, данных программы и данных о конфигурации Вашего CPU:

- CPU имеет в своем распоряжении EEPROM, в котором Вы можете постоянно хранить всю Вашу программу, некоторые области данных и данные о конфигурации CPU (см. рис. 6–10).
- CPU имеет в своем распоряжении мощный конденсатор, обеспечивающий надежность данных в ОЗУ также и после отключения источника питания CPU. В зависимости от варианта исполнения CPU мощный конденсатор может буферизовать ОЗУ в течение нескольких дней.
- Некоторые варианты CPU поддерживают поставляемый по желанию батарейный модуль, с помощью которого Вы можете продлить время, в течение которого буферизуется ОЗУ после отключения источника питания CPU. Этот батарейный модуль принимает на себя буферизацию данных после разряда мощного конденсатора.

В этом разделе объясняется, как CPU размещает в ОЗУ для хранения и возвращает оттуда обратно Ваши данные в различных ситуациях.

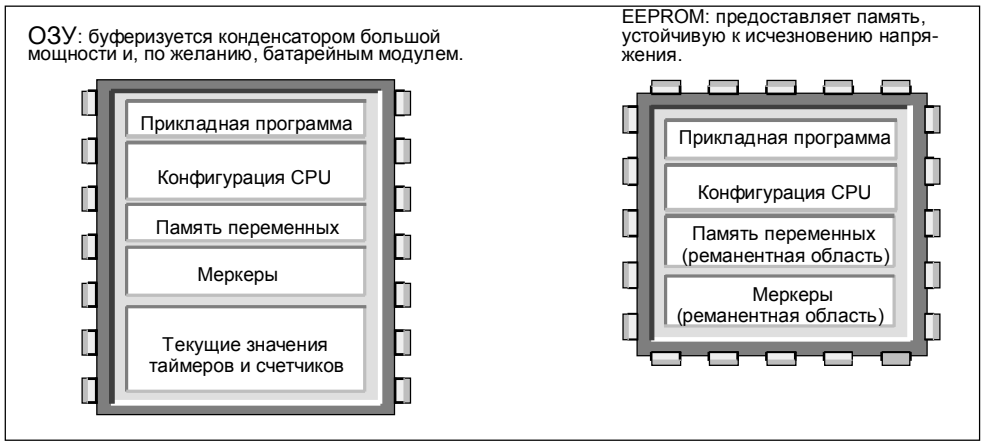

Рис. 6-10. Области памяти CPU S7-200

Загрузка Вашей программы в CPU и из CPU

Ваша программа состоит из трех компонентов: программы пользователя, блока данных (не обязательно) и конфигурации CPU (не обязательно). При загрузке программы в CPU эти компоненты помещаются в ОЗУ CPU (см. рис. 6–11). Кроме того, CPU автоматически копирует программу пользователя, блок данных (DB1) и конфигурацию CPU в EEPROM для постоянного хранения.

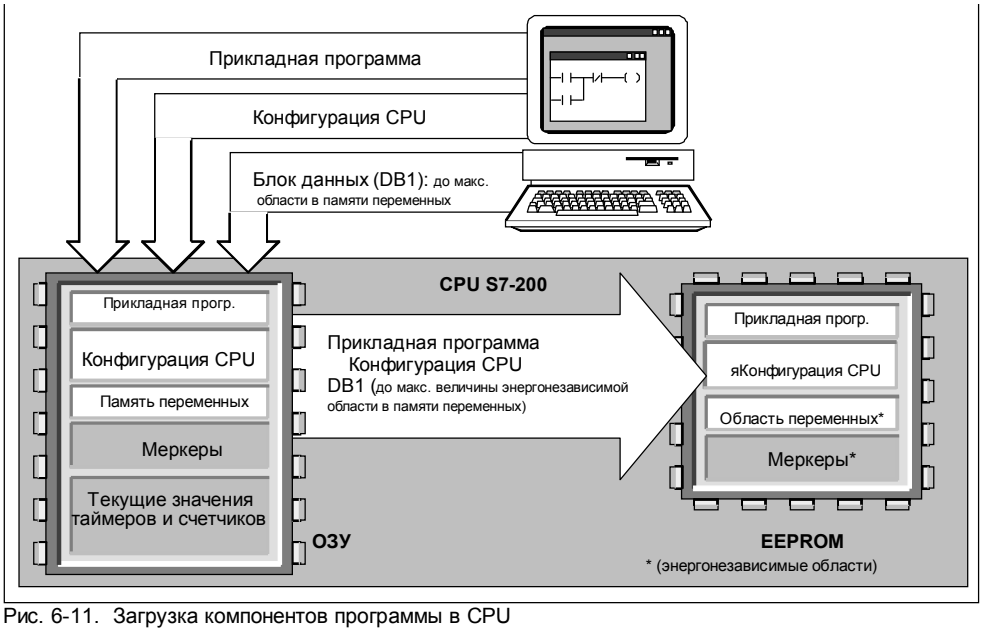

Рис. 6-11. Загрузка компонентов программы в CPU

Если Вы загружаете программу из CPU в PC или PG (см. рис. 6–12), то прикладная программа и конфигурация CPU загружаются из ОЗУ в Ваш компьютер. Если Вы загружаете из CPU блок данных, то энергонезависимая область блока данных (хранимая в EEPROM) сливается с возможно имеющимся остатком блока данных, который находится в ОЗУ. После этого компьютеру передается полный блок данных.

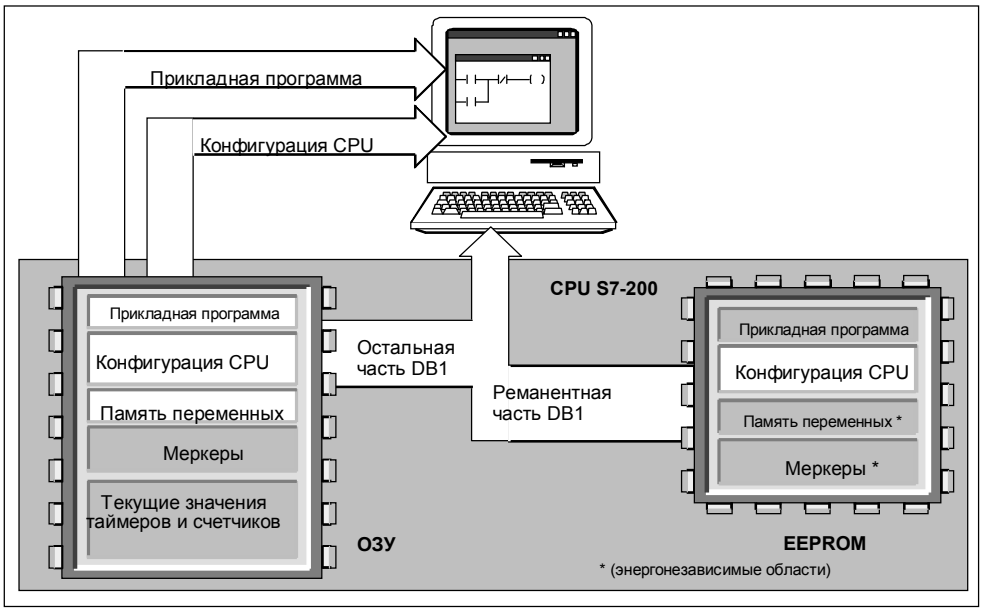

Рис. 6-12. Загрузка компонентов программы из CPU

Автоматическое сохранение меркеров (M) при потере напряжения

Первые 14 байтов из области памяти меркеров (с MB0 по MB13) при потере напряжения сохраняются в EEPROM, если перед этим они были определены как ременентные. CPU передает ременентные области меркеров в EEPROM
(см. рис. 6–13).

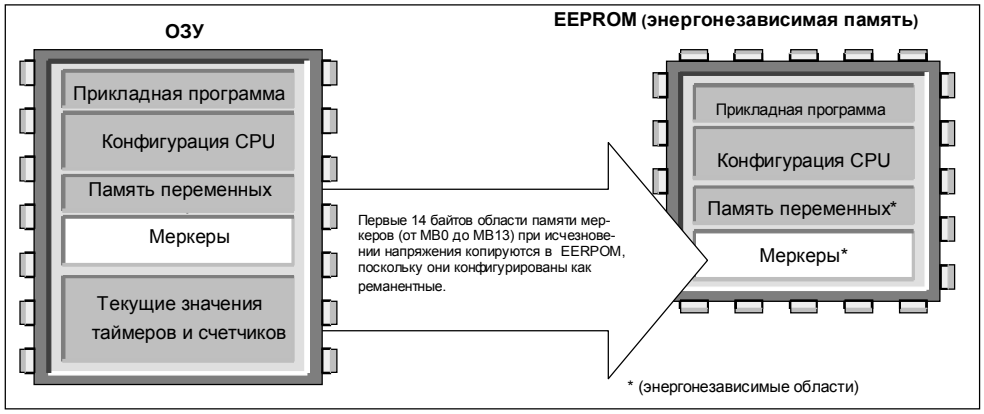

Рис. 6-13. Сохранение меркеров в EEPROM при исчезновении напряжения

Восстановление памяти при пуске

При включении напряжения питания CPU извлекает программу пользователя и конфигурацию CPU обратно из EEPROM (см. рис. 6–14).

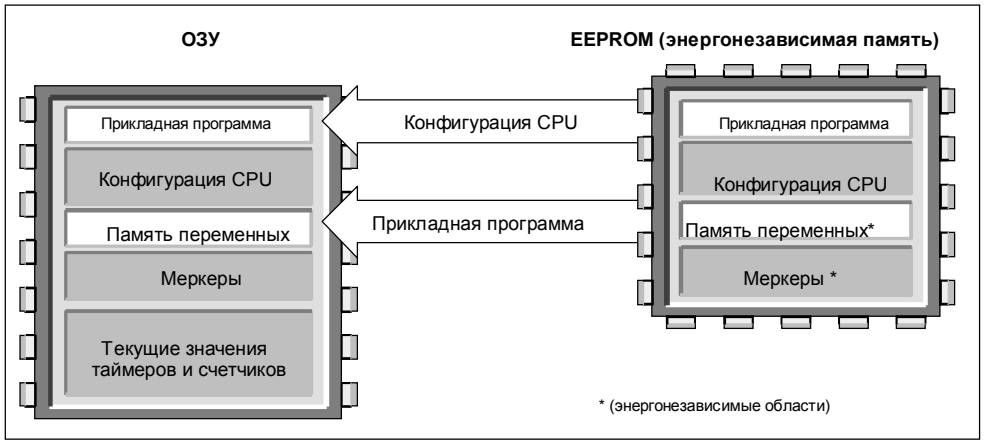

Рис. 6-14. Восстановление прикладной программы и конфигурации системы при пуске

При пуске CPU проверяет содержимое ОЗУ, чтобы проверить, выполнил ли мощный конденсатор буферизацию данных без ошибок. Если это имеет место, то ременентные области ОЗУ не изменяются. Нереманентные области памяти переменных восстанавливаются из соответствующей энергонезависимой области памяти переменных в EEPROM (см. рис. 6–15).

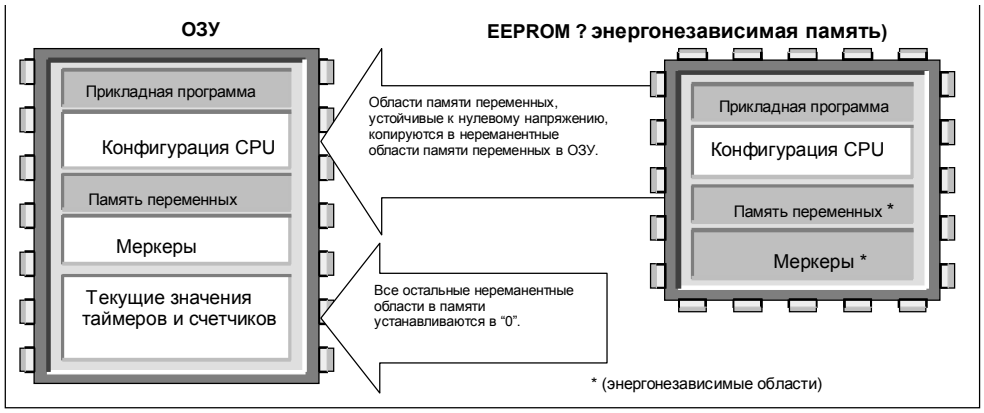

Рис. 6-15. Восстановление данных при пуске (данные были буферизованы в ОЗУ)

Если содержимое ОЗУ не могло быть буферизовано (например, вследствие крайне длительной потери напряжения), то CPU сбрасывает ОЗУ (а именно, ременантные и неременантные области) и в первом цикле после пуска устанавливает специальный меркер “Потеря ременантных данных” (SM0.2). Тогда данные из EEPROM копируются в ОЗУ (см. рис. 6–16).

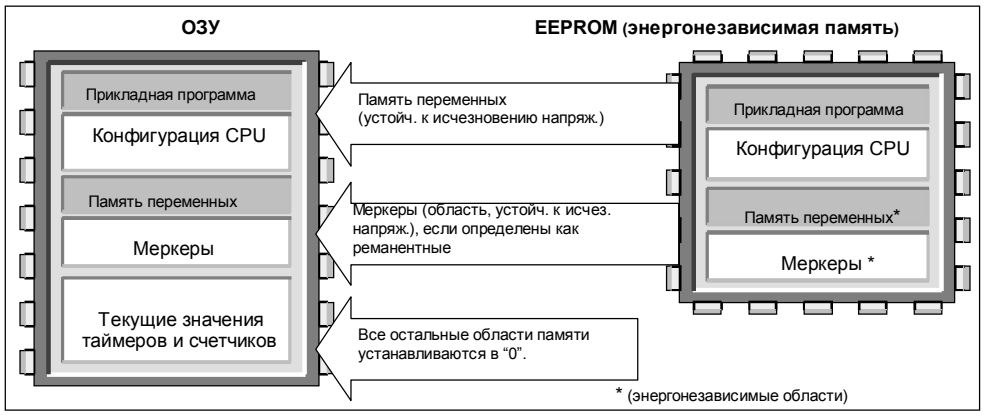

Рис. 6-16. Восстановление данных при пуске (данные не были буферизованы в ОЗУ)

Определение реманентных областей памяти

Вы можете определить в качестве реманентных максимум шесть областей и выбирать области памяти, которые хотите буферизовать в случае потери напряжения
(см. рис. 6–17). Вы можете задать в качестве реманентных определенные диапазоны адресов для следующих областей памяти: V, M, Z и T. В случае таймеров могут буферизоваться только реманентные таймеры (TONR).

Указание
У таймеров и счетчиков могут буферизоваться только текущие значения. Биты таймеров и счетчиков не являются реманентными.

Если Вы хотите задать в качестве реманентных определенные области памяти, то выберите пункт меню **CPU → Konfigurieren [CPU → Конфигурирование]**, а затем вкладку “Remanente Bereiche” [“Реманентные области”]. На рис. 6–17 показано диалоговое окно для задания реманентных областей. Если Вы хотите вызвать отображение предварительно установленных реманентных областей Вашего CPU, то выберите экранную кнопку ”Voreinstellungen” [“Предварительные установки”].

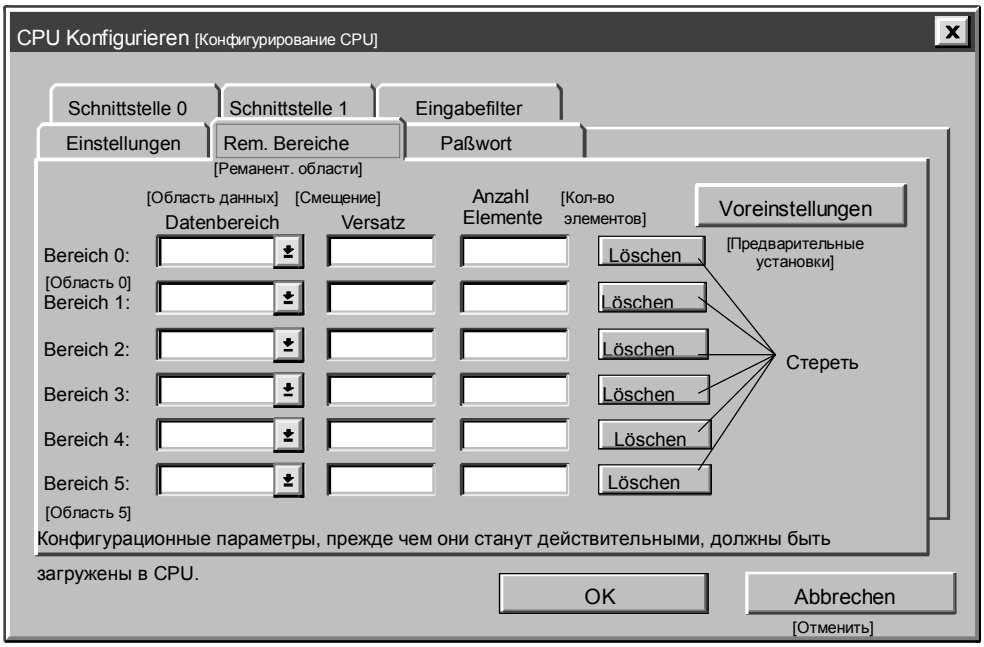

Рис. 6-17. Конфигурирование реманентных областей в памяти CPU

6.4 Использование программы для постоянного хранения данных

Вы можете сохранить в EEPROM значение (байт, слово или двойное слово), которое находится в памяти переменных. С помощью этой функции Вы можете сохранить любое значение из памяти переменных в постоянной энергонезависимой области памяти переменных.

Операция сохранения в EEPROM удлиняет время цикла на величину примерно от 15 мс до 20 мс. Значение, записываемое этой операцией, заменяет любое предыдущее значение, хранящееся по тому же адресу в памяти переменных EEPROM.

Указание
Операция сохранения в EEPROM не актуализирует данные в модуле памяти.

Копирование памяти переменных в EEPROM

Байт 31 специальных меркеров (SMB31) и слово 32 специальных меркеров (SMW32) дают CPU указание копировать значение из памяти переменных в постоянную область памяти переменных в EEPROM. На рис. 6–18 показан формат SMB31 и SMW32. Если Вы хотите запрограммировать CPU так, чтобы он записывал определенное значение в память переменных, действуйте следующим образом:

1. Загрузите в SMW32 адрес значения в памяти переменных, которое Вы хотите хранить устойчиво по отношению к исчезновению напряжения.
2. Загрузите размер данных в SM31.0 и SM31.1 (см. рис. 6–18.)
3. Установите SM31.7 в ”1”.

В конце каждого цикла CPU проверяет SM31.7. Если SM31.7 = 1, то указанное значение сохраняется в EEPROM. Эта операция заканчивается тогда, когда CPU сбрасывает SM31.7 в ”0”. Не изменяйте значение в памяти переменных до тех пор, пока не выполнится операция.

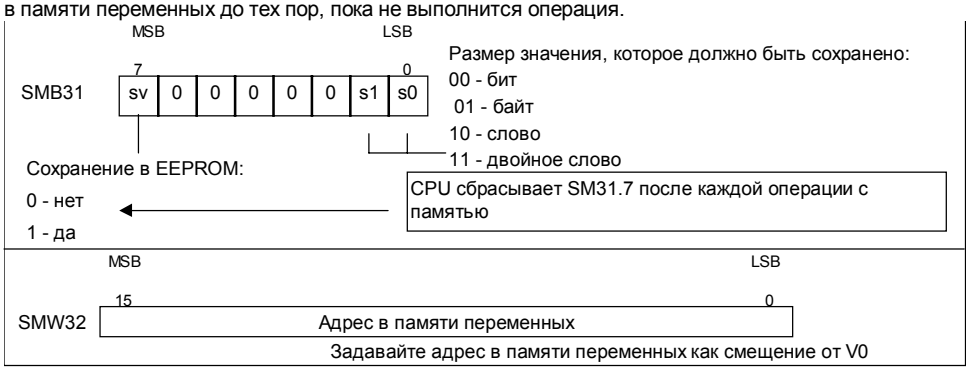

Рис. 6-18. Формат SMB31 и SMW32

Ограничения программируемого сохранения в EEPROM

Так как количество операций записи в память для EEPROM ограничено (минимум 100.000, в типичном случае 1.000.000), то Вы должны следить за тем, чтобы в EEPROM записывались только важные значения. В противном случае вследствие отказа EEPROM может выйти из строя также CPU. В типичной ситуации выполняйте операции записи в память для EEPROM только в случае особых событий. Такие события появляются довольно редко.
Например, если время цикла S7–200 составляет 50 мс и значение записывалось бы в EEPROM один раз за цикл, то EEPROM хватило бы по крайней мере на 5000 секунд, то есть менее полутора часов. Однако, если бы значение записывалось только один раз в час, то EEPROM выдержал бы по крайней мере уже 11 лет.

6.5 Сохранение Вашей программы в модуле памяти

Некоторые CPU поддерживают поставляемый по желанию модуль памяти, представляющий собой съемный EEPROM для Вашей программы. Вы можете использовать этот модуль памяти как дискету. CPU сохраняет в модуле памяти следующие компоненты:

- программу пользователя
- данные, хранящиеся в постоянной области памяти переменных EEPROM
- конфигурацию CPU

Копирование в модуль памяти

Вы можете копировать Вашу программу из ОЗУ в модуль памяти только в случае запуска CPU со вставленным модулем памяти.

Предостережение
Электростатические разряды могут повредить модуль памяти или предусмотренный для этого модуля разъем в CPU.

Когда Вы работаете с модулем памяти, Вы должны стоять на проводящем заземленном полу и/или надеть заземленный браслет. Модуль памяти нужно хранить в проводящем контейнере.

Вы можете вставлять и вытаскивать модуль памяти в то время, когда CPU включен. Для вставки модуля памяти снимите защитную ленту с разъема для модуля памяти. Разъем находится под откидной крышкой CPU. Затем вставьте модуль памяти в разъем. Модулю памяти придана такая форма, что он может вставляться в разъем только в одном направлении. Если Вы установили модуль памяти, то Вы можете копировать программу в этот модуль следующим образом:

1. Если Вы еще не загрузили программу в CPU, то загрузите ее сейчас.
2. Выберите команду меню **CPU** → **Speichermodul [CPU** → **Модуль памяти]**, чтобы копировать программу в модуль памяти. На рисунке 6–19 Вы видите компоненты памяти CPU, которые записываются в модуль памяти.
3. Вытащите модуль памяти из разъема (не обязательно).

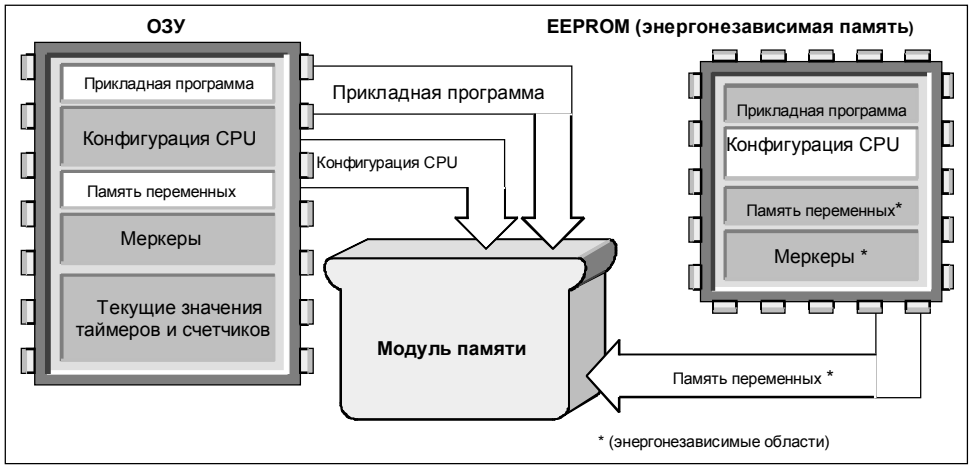

Рис. 6-19. Копирование памяти CPU в модуль памяти

Извлечение программы и содержимого памяти из модуля памяти

Если Вы хотите загрузить программу из модуля памяти в CPU, то Вы должны выключить CPU и снова включить со вставленным модулем памяти. Если модуль памяти вставлен, то CPU после включения выполняет следующие задачи
(см. рис. 6–20):

1. ОЗУ сбрасывается.
2. Содержимое модуля памяти копируется в ОЗУ.
3. Прикладная программа, конфигурация CPU и область памяти переменных (до максимального размера реманентной области в памяти переменных) копируются в постоянную память EEPROM.

Указание
Если Вы запускаете CPU со вставленным пустым модулем памяти, то возникает ошибка. Тогда вытащите модуль памяти и снова запустите CPU.

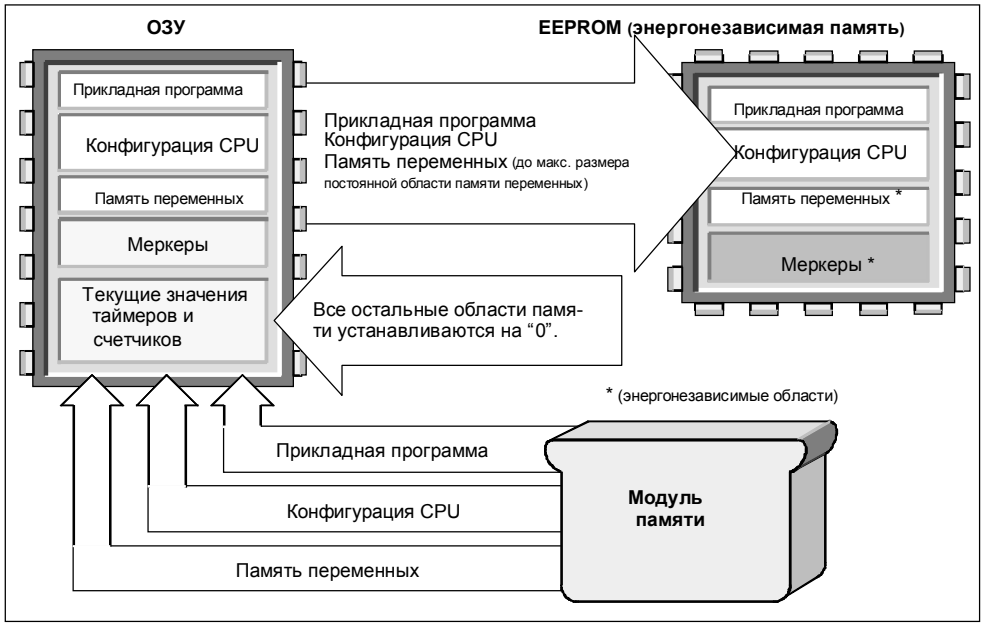

Рис. 6-20. Восстановление памяти при запуске (со вставленным модулем памяти)