

Обзор главы

В разделе	Вы найдете	на стр.
20.1	Вызов FC/SFC без параметров	20–2
20.2	Вызов FB, FC, системных FB и системных FC в виде блоков	20–4
20.3	Возврат	20–7
20.4	Функции Master Control Relay	20–8
20.5	Начало/конец Master Control Relay	20–9
20.6	Включение/выключение Master Control Relay	20–12

20.1. Вызов FC/SFC без параметров

Описание

С помощью операции *Вызов FC/SFC без параметров* Вы можете вызывать функцию (FC) или системную функцию (SFC), не имеющую параметров. В зависимости от предшествующего логического сопряжения речь идет об абсолютном или условном вызове (см. пример на рисунке 20–2).

При условном вызове Вы не можете задавать параметр с типом данных BLOCK_FC в качестве операнда в командной части функции (FC). Однако для функционального блока (FB) Вы можете задавать в качестве операнда параметр с типом данных BLOCK_FC.

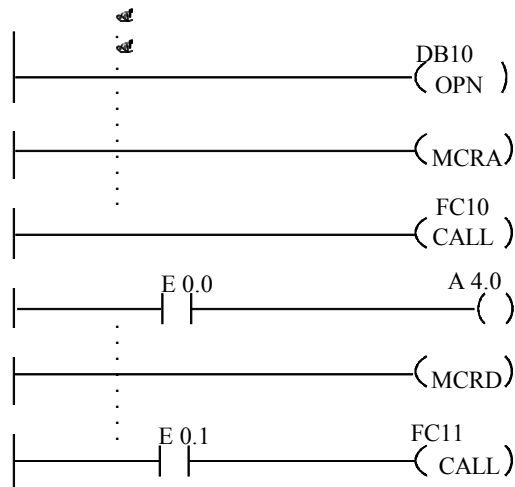
Абсолютный вызов выполняется только тогда, когда VKE равен "1". Если абсолютный вызов не выполнится, то после операции вызова VKE равен "0". Когда операция выполняется, она работает следующим образом:

- Она запоминает адрес возврата в вызывающий блок.
- Она запоминает два регистра блоков данных (блок данных и экземпляр блока данных).
- Она заменяет предыдущую область локальных данных текущей областью локальных данных.
- Она сдвигает МА–бит (бит активности MCR) в стек блоков (B–стек).
- Она создает новую область локальных данных для вызываемой FC или SFC.

После этого продолжается обработка программы в вызванном блоке. Информацию о передаче параметров Вы найдете в Руководстве по программированию /120/ .

Элемент KOP	Параметр	Тип данных	Область памяти	Описание
	Номер	BLOCK_FC	-	Номер FC или SFC ? напр., FC10 или SFC59). Какие SFC имеются в распоряжении, зависит от CPU. Условный вызов с параметром, имеющим тип данных BLOCK_FC, в качестве операнда возможен только в FB, но не в FC.

Рис. 20-1. Элемент “Вызов FC/SFC без параметров” и параметр



Если выполняется абсолютный вызов FC10, то операция CALL работает следующим образом:

- запоминает адрес возврата текущего FB;
- запоминает адреса для DB10 и экземпляра DB функционального блока;
- сдвигает МА-бит, установленный в “1” операцией MCRA, в стек блоков (В-стек) и сбрасывает его в “0” для вызванной функции FC10.

Выполнение программы продолжается в FC10. Если Вы хотите применить функцию MCR в FC10, то Вы должны ее там вновь активизировать. Когда FC10 завершена, обработка программы продолжается в вызывающем FB. Бит МА восстанавливается. DB10 и экземпляр DB определенного пользователем FB вновь становятся текущими DB, независимо от того, какие DB использовались в FC10.

После возврата из FC 10 состояние сигнала E 0.0 присваивается выходу A 4.0. При вызове FC11 речь идет об условном вызове, который выполняется только тогда, когда вход E 0.1 = 1. Если вызов выполняется, функционирование происходит так же, как и при вызове FC10.

Запись битов в слове состояния

Абсолютный вызов		BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
записывает	-	-	-	-	0	0	1	-	0	
Условный вызов		BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
записывает	-	-	-	-	0	0	1	1	0	

Рис. 20-2. Вызов FC/SFC без параметров

20.2. Вызов FB, FC, системных FB и системных FC в виде блоков

Описание

Вы можете вызывать функциональные блоки (FB), функции (FC), системные функциональные блоки (SFB) и системные функции (SFC), выбирая их в блоке выбора. Вы найдете их в конце списка семейств операций под следующими именами:

- FB Bausteine (блоки FB)
- FC Bausteine (блоки FC)
- SFB Bausteine (блоки SFB)
- SFC Bausteine (блоки SFC)

Когда Вы выбираете один из этих блоков, на Вашем экране появляется блок с номером или символическим именем функции или функционального блока и с соответствующими параметрами.

Блок, который Вы желаете вызвать, должен быть скомпилированным и присутствовать в вашем программном файле или в CPU.

Когда операция (вызов FB, FC, системного FB и системной FC в виде блока) выполняется, она работает следующим образом:

- Она запоминает адрес возврата в вызывающий блок.
- Она запоминает два регистра блоков данных (блок данных и экземпляр блока данных).
- Она заменяет предыдущую область локальных данных текущей областью локальных данных.
- Она сдвигает MA-бит (бит активности MCR) в стек блоков (B-стек).
- Она создает новую область локальных данных для вызываемой FC или SFC.

Указание

Когда DB- и DI-регистры запоминаются, может оказаться так, что они указывают не на блоки данных, которые Вы открыли. По причине механизмов копирования при передаче параметров, особенно в случае функциональных блоков, DB-регрстр иногда переписывается при компиляции. Более подробную информацию по этому поводу возьмите из Руководства по программированию /234/.

После этого продолжается обработка программы в вызванном блоке.

**Разрешающий
выход**

Разрешающий выход (ENO) KOP-блока соответствует BIE-биту слова состояния (см. главу 6.3). Когда Вы пишете функциональный блок или функцию, которую Вы хотели бы вызывать из KOP, то независимо от того, пишете ли Вы FB или FC в форме AWL или KOP, Вы должны учитывать BIE-бит. Вы сохраняете VKE в BIE-бите посредством операции SAVE (в AWL) или посредством катушки ---(SAVE) (в KOP) согласно следующим критериям:

- Сохраняйте VKE "1" в BIE-бите в том случае, когда FB или FC обрабатывается без ошибки.
- Сохраняйте VKE "0" в BIE-бите в том случае, когда при обработке FB или FC появляется ошибка.

Программируйте эти операции в конце FB или FC, так чтобы эти операции обрабатывались как последние операции в блоке.



ошибки сохраняйте VKE в конце FB или FC так, как описано выше.

Предупреждение

BIE-бит может неумышленно сбрасываться в "0".
Если Вы пишете FB или FC в KOP и не обрабатываете BIE-бит так, как описано выше, то FB или FC может переписать BIE-бит другого FB или другой FC. Во избежание такой ошибки сохраняйте VKE в конце FB или FC так, как описано выше.

**Воздействие
вызова на биты
слова состояния**

Рис. 20–3 описывает воздействие условного или абсолютного вызова блока на биты слова состояния (см. главу 6.3).

		BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Условный:	записывает	x	-	-	-	0	0	*	x	x
Абсолютный:	записывает	-	-	-	-	0	0	x	x	x

Рис. 20-3. Влияние вызова блока на биты в слове состояния

Параметры

Параметры вызываемого блока представляются в KOP-блоке. В зависимости от вида блока при назначении параметров нужно поступать по-разному.

- Для функции (FC) Вы должны все формальные параметры снабдить фактическими параметрами.
- В случае функциональных блоков (FB) задание фактических параметров является необязательным. Однако Вы должны снабдить вызов FB экземпляром блока данных (экземпляр DB). Если формальный параметр не был снабжен фактическим параметром, то FB работает со значениями, имеющимися в его экземпляре DB.

Структурированным INOUT-параметрам, а также параметрам типов "указатель" [Pointer] и "массив" [Array] Вы должны предоставить в распоряжение фактический параметр (по крайней мере, при первом вызове).

Каждый фактический параметр, предоставляемый функциональному блоку в распоряжение при вызове, должен иметь такой же тип данных, как его формальный параметр.

Информацию о том, каким образом Вы можете программировать функцию или работать с его параметрами, возьмите из Руководства по программированию /234/.

Таблица показывает графический блок для вызова FB, FC, SFB и SFC и описывает параметры, которые являются общими для графического блока всех этих функциональных блоков. Если Вы вызываете блок из окна диалога, то номер блока автоматически отображается в заголовке блока (номер FB, FC, SFB или SFC, например, FC10).

Блок KOP	Параметр	Тип данных	Область памяти	Описание
DB-Nr. Блок № EN ENO IN OUT IN/OUT	DB-Nr.	BLOCK DB	-	Номер DB. Эти данные нужно указывать только тогда, когда Вы хотите вызвать FB.
	EN	BOOL	E, A, M, D, L	Разрешающий вход
	ENO	BOOL	E, A, M, D, L	Разрешающий выход

Рис. 20-4. Блок и параметры для вызова FB, FC, SFB и SFC



Рис. 20-5. Вызов FB как блока

20.3 Возврат

Описание С помощью операции *Возврат* Вы можете покидать блоки. Вы можете покидать блок условно. Операция *Возврат* сохраняет VKE условия в BIE-бите слова состояния. Если блок завершается на основании условного возврата, то состояние сигнала BIE-бита в блоке, на который возвращается обработка программы, равно "1".

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
—(RET)	Нет	-	-	-

Рис. 20-6. Элемент "Возврат"

E 0.0

(RET)

Блок покидается, если E 0.0 = 1. Тогда бит BIE имеет то же состояние сигнала, что и вход E 0.0 (= 1).

Запись битов в слове состояния

Условный возврат (возврат, если VKE = 1)

	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
записывает	x	-	-	x	0	1	1	0	

Рис. 20-7. Возврат

20.4. Функции Master Control Relay

Определение Master Control Relay

Master Control Relay (MCR) (MCR, см. также главу 20.5) используется в релейно-контактных схемах для активизации и деактивизации потока сигналов (цепи тока). Деактивизированная цепь тока соответствует последовательности операций, записывающей нулевое значение вместо расчетного значения, или последовательности операций, которая оставляет неизменной существующее значение в памяти. Операции, которые запускаются одной из операций, показанных на рисунке 20–8, зависят от MCR.

Операция *Катушка реле, выход* и операция *Коннектор* записывают в память “0”, если MCR равен ”0”. Операции *Установка выхода* и *Сброс выхода* не изменяют существующее значение (см. рис. 20–9).


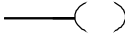
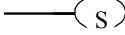
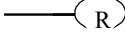
Символ или имя в блоке	Описание	Глава в данном руководстве
	Коннектор	8.5
	Катушка реле, выход	8.4
	Установка выхода	8.8
	Сброс выхода	8.9
SR	Установка-сброс триггера	8.22
RS	Сброс-установка триггера	8.23
MOVE	Передача значения	14.1

Рис. 20-8. Операции, на которые влияют зоны MCR

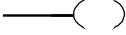
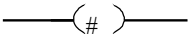
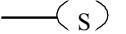
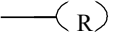
Состояние сигнала MCR	Катушка реле, выход  	Установка выхода или Сброс выхода   SR RS	T MOVE
0	Записывает "0" (Имитирует реле, которое при исчезновении напряжения переходит в состояние покоя)	Не записывает (Имитирует реле, которое при исчезновении напряжения остается в текущем состоянии)	Записывает “0” (Имитирует компонент, который при исчезновении напряжения дает значение “0”)
1	Нормальная обработка	Нормальная обработка	Нормальная обработка

Рис. 20-9. Операции, зависящие от MCR, и их реакция на состояние сигнала MCR

20.5 Начало/конец Master Control Relay

Начало MCR

С помощью операции *Начало Master Control Relay* Вы включаете MCR–зависимость последующих команд. После этой команды Вы можете с помощью команд *Включение Master Control Relay* и *Выключение Master Control Relay* программировать MCR–зоны (см. главу 20.6). Когда Ваша программа активизирует MCR–область, все MCR–действия зависят от содержимого MCR–стека (см. рис. В–4).


Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	Нет	-	-	Активизирует функцию MCR

Рис. 20-10. Элемент “Начало Master Control Relay”

Конец MCR

С помощью операции *Конец Master Control Relay* Вы выключаете MCR–зависимость последующих команд. После этой команды Вы больше не можете программировать MCR–зоны. Когда Ваша программа деактивирует MCR–область, MCR всегда является токоведущим, независимо от записей в MCR–стеке.


Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	Нет	-	-	Деактивирует функцию MCR

Рис. 20-11. Элемент “Конец Master Control Relay”

MCR–стек и бит, управляющий его зависимостью (МА–бит), относятся к соответствующему уровню и должны сохраняться и извлекаться каждый раз, когда происходит переключение на последующий уровень. В начале каждого последующего уровня они предварительно устанавливаются (входные биты MCR с 1 по 8 устанавливаются в ”1”, указатель стека MCR = 0 и МА–бит = 0).

MCR–стек передается от блока к блоку, а МА–бит при каждом вызове блока сохраняется и сбрасывается в ”0”. В конце блока он восстанавливается.

MCR может быть реализован так, что он оптимизирует время работы CPU, генерирующих код. Основанием для этого является то, что зависимость от MCR не передается дальше к блоку, а должна активизироваться явно посредством MCRA–операции. CPU, генерирующий код, распознает эту операцию и генерирует дополнительный код, необходимый для использования MCR–стека, до тех пор, пока он не обнаружит MCRD–операцию или пока не будет достигнут конец блока. Это не увеличивает времени исполнения для операций вне MCRA/MCRD–области.

Вы можете использовать операции MCRA и MCRD в своей программе всегда только в паре.

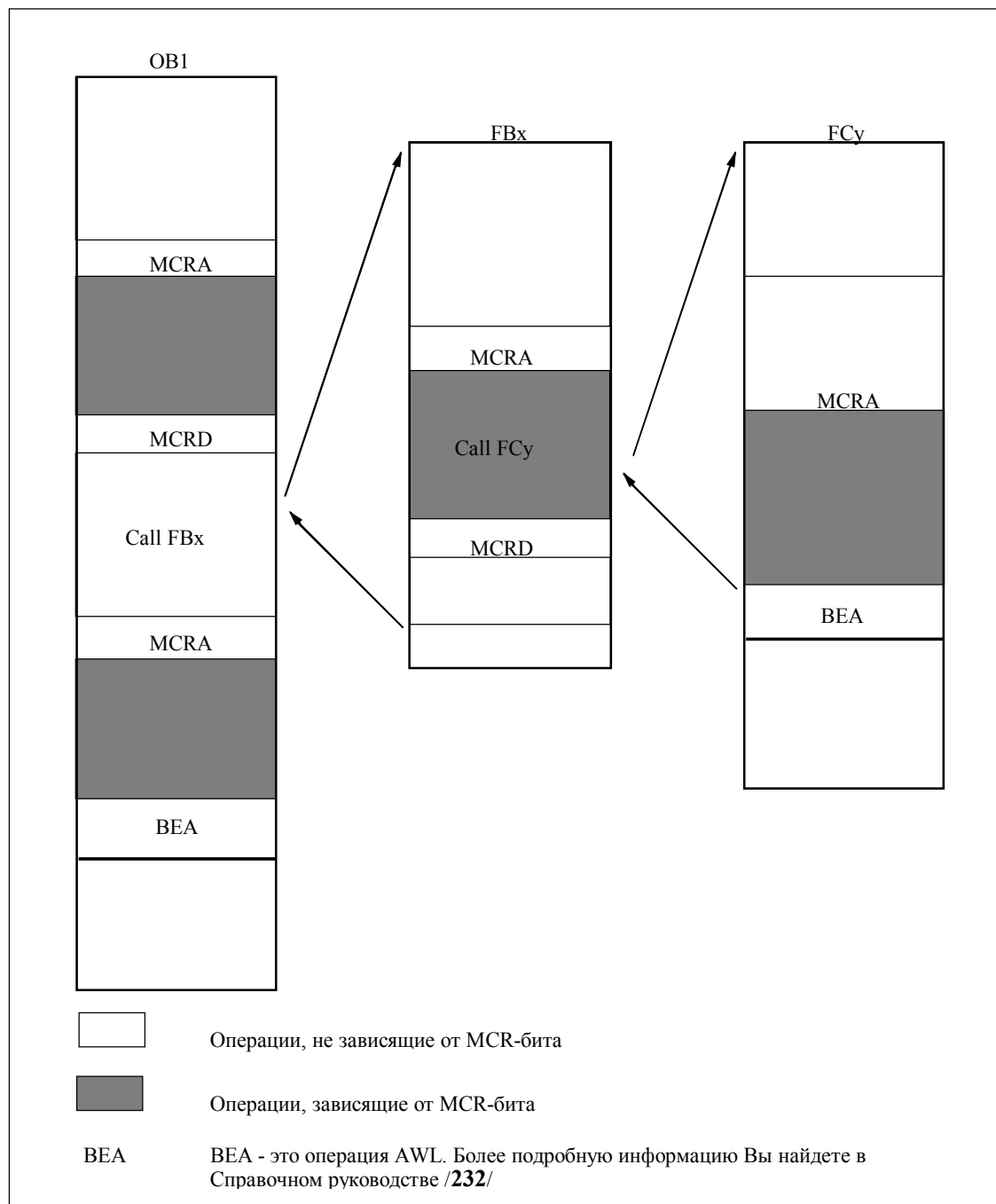
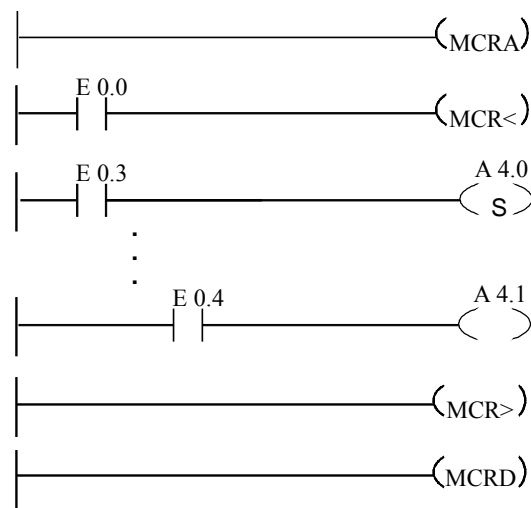


Рис. 20-12. Активирование и деактивирование области MCR

Операции, запрограммированные между MCRA и MCRD, зависят от состояния сигнала MCR-бита. Операции, запрограммированные вне MCRA–MCRD–последовательности, не зависят от состояния сигнала MCR-бита. Если MCRD–операция отсутствует, то операции, запрограммированные между MCRA и BEA, зависят от MCR-бита. (BEA является AWL–операцией. Более подробную информацию Вы найдете в Справочном руководстве /232/.)



Операция --(MCRA) активизирует функцию MCR до следующего MCRD. Операции между --(MCR<) и --(MCR>) обрабатываются в зависимости от МА-бита (здесь E0.0):

- Если E 0.0 = 1, то:
 - A 4.0 устанавливается в "1" при E 0.3 = 1
 - A 4.0 не меняется при E 0.3 = 0
 - состояние сигнала E 0.4 присваивается выходу A 4.1.
- Если E 0.0 = 0:
 - A 4.0 не меняется независимо от состояния сигнала на E 0.3
 - A 4.1 имеет значение "0" независимо от состояния сигнала на E 0.4.

Запись битов в слове состояния										
записывает	-	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
	-	-	-	-	-	-	-	-	-	

Рис. 20-13. Начало и конец Master Control Relay

Вы должны сами программировать зависимость функций (FC) и функциональных блоков (FB) от MCR в этих блоках. Если такая функция или такой функциональный блок вызывается изнутри MCRA–MCRD–последовательности, то все команды внутри этой последовательности не зависят автоматически от MCR–бита. Для этого используйте операцию MCRA в вызываемом блоке.

Предупреждение



Риск ущерба для персонала и оборудования:
 Никогда не используйте операцию MCR в качестве АВАРИЙНОГО ВЫКЛЮЧАТЕЛЯ или предохранительного устройства для персонала. MCR не является заменой жестко коммутированного Master Control Relay.

20.6. Включение/выключение Master Control Relay

Включение MCR С помощью операции *Включение Master Control Relay* (MCR<) VKE сохраняется в MCR-стеке и открывается MCR-зона. Приведенные в таблице операции испытывают внутри этой зоны влияние VKE, сохраненного в MCR-стеке при открытии MCR-зоны. MCR-стек работает как LIFO-буфер (LIFO = last in, first out), который может содержать максимум 8 записей. Если стек заполнен, то операция *Включение Master Control Relay* порождает сообщение об ошибке MCR-стека (MCRF).

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	Нет	-	-	Открывает зону MCR

Рис. 20-14. Элемент “Включение Master Control Relay”

Выключение MCR С помощью операции *Выключение Master Control Relay* (MCR>) Вы закрываете последнюю открытую MCR-зону. Это происходит путем стирания VKE-записи из MCR-стека, куда она была помещена операцией *Включение Master Control Relay*. MCR-стек работает как LIFO-буфер (LIFO = last in, first out [последним пришел, первым вышел]). Запись, освобождающаяся в конце этого стека, устанавливается в ”1”. Если стек уже пустой, то операция *Выключение Master Control Relay* порождает сообщение об ошибке MCR-стека (MCRF).



Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	Нет	-		Закрывает последнюю открытую зону MCR

Рис. 20-15. Элемент “Выключение Master Control Relay”

MCR управляется стеком, который имеет ширину один бит и глубину восемь записей (см. рис. 20-16). MCR является активным до тех пор, пока все восемь записей в стеке не станут равны ”1”. Операция --(MCR<) копирует результат логического сопряжения в MCR-стек. Операция --(MCR>) удаляет последнюю запись из стека и устанавливает освобождающийся адрес стека в “1”. Если возникает ошибка, например, если друг за другом подряд следуют более восьми операций --(MCR>) или делается попытка выполнить операцию --(MCR>) при пустом MCR-стеке, то такая ошибка порождает сообщение об ошибке MCRF. Контроль MCR-стека выполняет указатель стека (MSP: 0 = пусто, 1 = одна запись, 2 = две записи, ..., 8 = восемь записей).

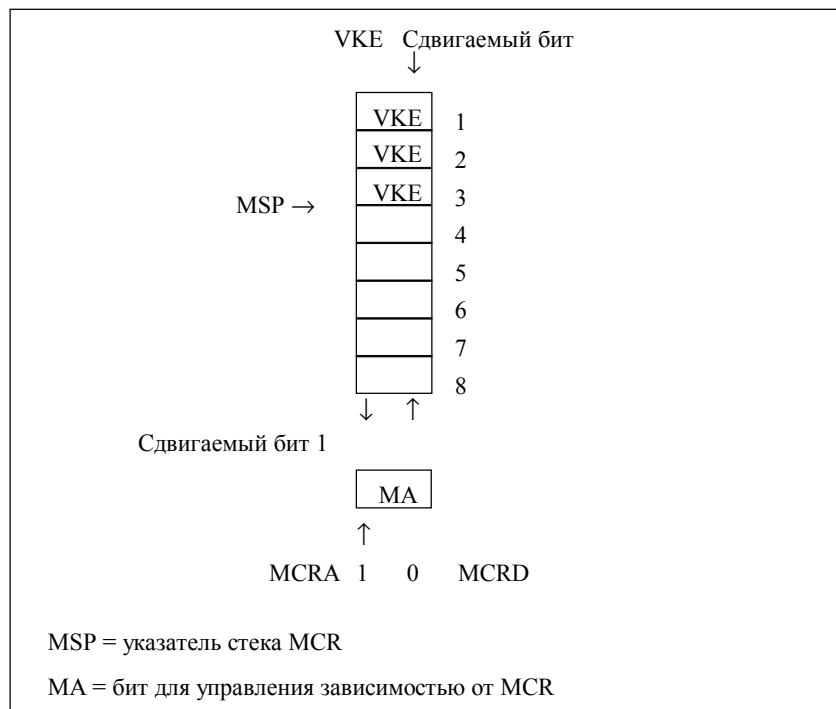


Рис. 20-16. Стек Master Control Relay

Вы можете использовать в своей программе операции `--(MCR<)` и `--(MCR>)` всегда только в паре.

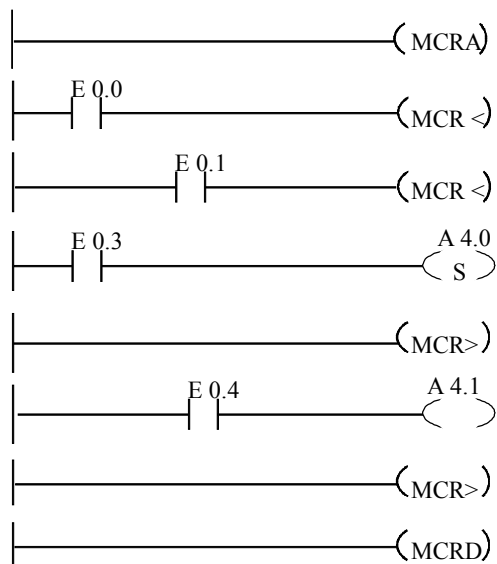
Операция `--(MCR<)` принимает состояние сигнала VKE и копирует его в MCR-бит.

Операция `--(MCR>)` устанавливает MCR-бит абсолютно в "1". Вследствие такого свойства каждая последующая операция между операциями `--(MCRA)` и `--(MCRD)` работает независимо от MCR-бита (по поводу `--(MCRA)` и `--(MCRD)`: см. подробности выше).

Вложение операций `--(MCR<)` и `--(MCR>)` Вы можете вкладывать операции `--(MCR<)` и `--(MCR>)` попарно друг в друга. Максимальная глубина вложения равна восьми, то есть Вы можете записывать друг за другом максимум восемь операций `--(MCR<)`, прежде чем вставить операцию `(MCR>)`. Вы должны программировать одинаковое число операций `--(MCR<)` и `--(MCR>)`.

Если операции `--(MCR<)` вкладываются, то образуется MCR-бит более низкого уровня. После этого операция `--(MCR<)` логически связывает текущий VKE с текущим MCR-битом в соответствии с таблицей истинности операции И.

Когда операция `--(MCR>)` завершает один уровень вложения, она берет MCR-бит более высокого уровня.



Когда операция --(MCRA) активизирует функцию MCR, Вы можете создать до восьми вложенных зон MCR. В нашем примере имеется две зоны MCR. Первая операция --(MCR>) работает совместно со второй операцией --(MCR<). Все операции между второй группой скобок MCR (MCR<MCR>) относятся ко второй зоне MCR. Эти операции выполняются следующим образом:

- E 0.0 = 1: состояние сигнала E 0.4 присваивается выходу A 4.1.
- E 0.0 = 0: выход A 4.1 равен "0", независимо от состояния сигнала на E 0.4
- Выход A 4.0 не меняется, независимо от состояния сигнала на E 0.3.
- E 0.0 и E 0.1 = 1: выход A 4.0 устанавливается в "1", если E 0.3 = 1, и A 4.1 = E 0.4.
- E 0.1 = 0: выход A 4.0 не меняется, независимо от состояния сигнала на E 0.3 и E 0.0.

Запись битов в слове состояния

записывает	-	BIE	-	A1	-	A0	-	OV	-	OS	0	OR	1	STA	-	VKE	0	/ER
------------	---	-----	---	----	---	----	---	----	---	----	---	----	---	-----	---	-----	---	-----

Рис. 20-17. Включение и выключение Master Control Relay