

Обзор главы

В разделе	Вы найдете	на стр.
23.1	Параметризация при вызове FC и FB	23–2
23.2	Вызов функций и функциональных блоков с помощью CALL	23–3
23.3	Вызов функций и функциональных блоков с помощью CC и UC	23–7
23.4	Функции Master Control Relay	23–10
23.5	Операции для Master Control Relay	23–11
23.6	Конец блока	23–15

23.1. Параметризация при вызове FC и FB

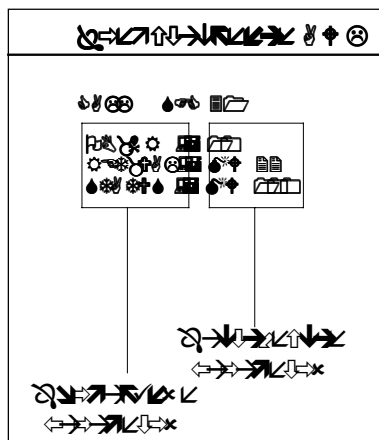
Понятия

Для вызова параметризуемых блоков важны два понятия: **формальные параметры** и **фактические параметры**.

Формальный параметр - это параметр, имя и тип данных которого были определены при создании параметризуемого блока и описаны, например, как INPUT или OUTPUT. При вызове блока (напр., CALL SFC 31) в инкрементном редакторе STEP 7 автоматически отображает список формальных параметров.

Этим формальным параметрам Вы затем должны поставит в соответствие **фактические параметры**. Фактический параметр - это параметр, со значением которого работает функция или функциональный блок во время выполнения программы.

На следующем рисунке показан вызов SFC 31 "QRY_TINT" (опрос прерывания по времени) на AWL.



23.2. Вызов функций и функциональных блоков с помощью CALL

Описание

С помощью операции вызова блоков (CALL) Вы можете вызывать функции (FC) и функциональные блоки (FB), которые Вы сами создали для своей программы или получили от фирмы Siemens в качестве стандартных функций или функциональных блоков. Операция CALL вызывает FC или FB, которую или который Вы указали в качестве операнда, независимо от результата логической операции или любого другого условия.

Когда Вы вызываете с помощью операции CALL функциональный блок, Вы должны снабдить его экземпляром блока данных (экземпляром DB) или описать как локальный экземпляр. В этом экземпляре блока данных сохраняются все статические переменные и фактические параметры функционального блока.

Информацию о том, как можно запрограммировать функцию или функциональный блок или как работать с их параметрами, Вы найдете в Руководстве по программированию /234/.

Формальные и фактические параметры

При вызове функции (FC) или функционального блока (FB) Вы должны присвоить формальным параметрам, определенным при описании блока, соответствующие фактические параметры.

Фактические параметры, указываемые при вызове функционального блока, должны иметь тот же тип данных, что и соответствующие формальные параметры.

Задание фактических параметров

Фактические параметры, используемые при вызове функции или функционального блока, задаются, как правило, в виде символических имен. Абсолютная адресация фактических возможна только у операндов, максимальная величина которых составляет двойное слово (напр., E 1.0, MB2, AW4, ED0).

При вызове функций все формальные параметры должны быть снабжены фактическими. При вызове функциональных блоков необходимо задавать только те фактические параметры, которые должны были измениться по сравнению с последним вызовом (фактические параметры после обработки FB сохраняются в экземпляре блока данных).

При вызове функционального блока операция CALL копирует в экземпляр блока данных функционального блока одно из следующих значений в зависимости от типа данных фактического параметра и описания формального параметра (ВХОД, ВЫХОД, ВХОД-ВЫХОД).

- Значение фактического параметра
- Указатель на операнд фактического параметра
- Указатель на L-стек вызывающего блока, в котором было буферизовано значение фактического параметра.

**Вызов FB
с экземпляром DB
и параметрами
блока**

Вызов происходит при задании

- имени функционального блока,
- имени экземпляра блока данных, а также
- параметров.

Вызов может выполняться абсолютно или символически.

Абсолютный вызов:
CALL FBx,DBu (*передача параметров*);
x = номер блока
y = номер блока данных

Символический вызов:
CALL имя_fb, имя_блока_данных (*передача параметров*);
имя_fb = символическое имя блока
имя_блока_данных = символическое имя блока данных

Примеры

В следующем примере программы вызывается функциональный блок FB40 с экземпляром блока данных DB41. В этом примере формальные параметры имеют следующие типы данных:

EIN1: BOOL
EIN2: WORD
AUS1: DWORD

AWL	Объяснение
CALL FB40,DB41	Вызов функционального блока FB40 с экземпляром блока данных DB41.
EIN1:= E 1.0	EIN1 (формальный параметр) ставится в соответствие E 1.0 (фактическому параметру).
EIN2:= MW2	EIN2 (формальный параметр) ставится в соответствие MW2 (фактическому параметру).
AUS1:= MD20	AUS1 (формальный параметр) ставится в соответствие MD20 (фактическому параметру).
L MD20	С помощью этой операции программа обращается к формальному параметру AUS1.

В следующем примере вызывается функциональный блок FB50 с экземпляром блока данных DB51. В этом примере формальные параметры имеют следующие типы данных:

EIN10: BOOL
AUS11: STRUCT
V1: BOOL
V2: INT
END_STRUCT

AWL	Объяснение
CALL FB50,DB51	Вызов функционального блока FB50 с экземпляром блока данных DB51.
EIN10:= E1.0	EIN10 (формальный параметр) ставится в соответствие E 1.0 (фактическому параметру).
AUS11:= АКТРА11	В этом случае невозможно задать абсолютный фактический параметр (напр., MW10), так как формальный параметр AUS11 был определен как структура. Вместо этого был задан символический фактический параметр АКТРА11. Обратите, пожалуйста, внимание, что АКТРА11 должен иметь ту же структуру, что и формальный параметр AUS11.

Доступ к значению структуры AUS11 в FB50 происходит следующим образом:

AWL	Объяснение
U AUS11.V1	Выполнить логическую операцию И с битом AUS11.V1.
L AUS11.V2	Загрузить AUS11.V2 в АККУ 1.

Вызов мультиэкземпляров

Вызов происходит при задании

- имени экземпляра (= имени статической переменной типа FB z), а также
- параметров.

Вызов всегда символический.

CALL *имя_экземпляра (передача параметров);*

AWL	Объяснение
FUNCTION_BLOCK FB 11	Исходный файл
VAR	
loc_inst : FB 10;	Описание мультиэкземпляра с типом данных FB 10
END_VAR	
BEGIN	
NETWORK	
CALL #loc_inst (Вызов мультиэкземпляра в соответствии с синтаксисом
in_bool := M 0.0);	Передача параметров (при этом in_bool является переменной, описанной в FB 10)

Вызов FC с параметрами блока

Вызов производится при задании

- имени функции, а также
- параметров.

Вызов может производиться абсолютно или символически

Абсолютный вызов:

CALL FCx (*передача параметров*);

x = номер блока

Символический вызов:

CALL имя_fc (*передача параметров*);

имя_fc = символическое имя блока

Пример

В следующем примере программы вызывается функция FC80 с параметрами блока. В этом примере формальные параметры имеют следующие типы данных:

INK1: BOOL
INK2: INT
AUS: WORD

AWL	Объяснение
CALL FC80 INK1:= M 1.0	Вызов функции FC80. INK1 (формальный параметр) ставится в соответствие M 1.0 (фактическому параметру).
INK2:= EW2	INK2 (формальный параметр) ставится в соответствие EW2 (фактическому параметру).
AUS:= AW4	AUS (формальный параметр) ставится в соответствие AW4 (фактическому параметру).

Вызов FC, возвращающего значение

Вы можете разработать функцию (FC), возвращающую значение (RET_VAL). Если, например, Вам хочется разработать арифметическую операцию с вещественными числами, то Вы можете использовать это возвращаемое значение использовать как выход для результата функции. Когда Вы вызываете функцию в своей программе, то Вы предоставляете в распоряжение выход RET_VAL с адресом двойного слова, так что он может воспринять 32-битный результат арифметической операции.

23.3. Вызов функций и функциональных блоков с помощью CC и UC

Описание

С помощью следующих операций Вы можете вызывать разработанный Вами для своей программы функции (FC) так же, как и с помощью операции CALL. Однако, Вы не можете передавать параметры.

- Условный вызов блока (CC): Эта операция вызывает функцию, указанную в качестве операнда. Вызов производится только тогда, когда результат логической операции равен "1".
- Безусловный вызов блока (UC): Эта операция вызывает функцию, указанную в качестве операнда. Вызов выполняется независимо от результата логической операции или какого-либо иного условия.

При вызове функционального блока (FB) командой CC или UC операнду не может быть поставлен в соответствие никакой блок данных.

Формат адресации

Операции CC и UC могут вызывать функции (FC) посредством прямой или косвенной через память адресации или через FC, передаваемые в качестве параметров (см. таблицы 23–1 и 23–2). Операндом является FC плюс указание номера функции.

Таблица 23–1. Операнды операций CC и UC: прямая и косвенная адресация

FC– или FB–часть операнда	Максимальная адресная область в зависимости от вида адресации	
	прямая	косвенная через память
FC FB	от 0 до 65 535	[DBW] от 0 до 65 534 [DIW] [LW] [MW]

Таблица 23–2. Операнды операций CC и UC: FC передается как параметр

Операнд	Виды адресных параметров
Имя формального параметра или символическое имя	BLOCK_FC ¹

¹ Параметры типа BLOCK_FC не могут применяться в FC с операцией CC.

Пример

Разработанную Вами FC, которой Вы, например, дали номер 12, Вы можете вызвать одной из следующих двух операций. Какую операцию Вы выберете, зависит от того, хотите ли Вы выполнить условный или безусловный вызов:

CC FC12 (Вызвать FC12, если VKE равно "1")
UC FC12 (вызвать FC12 независимо от VKE)

Назначение фактических параметров

В зависимости от типа данных у Вас есть различные возможности назначения формальному параметру фактического параметра при вызове функции или функционального блока. Следующая таблица упорядочена по длине типа данных:

Таблица 23–3. Назначение фактических параметров

Формальный параметр типа ...	Пример назначения фактического параметра		
	Прямой ввод (значение)	Ввод элемента глобальных данных	Символический ввод ¹
BOOL (бит)	TRUE	M 100.0 E 0.0 A 0.0 DBX 3.0	#OK_MERKER
BYTE (байт)	B#16#1F	MB 100 EB 0 AB 0	#TYP_BYTE
CHAR	'K'	DBB 1	#TYP_CHAR
WORD (слово)	W#16#1F12 2#0001_1111_0001_0010 C#32 B#(5,25)	MW 100 EW 0 AW 0 DBW 2	#TYP_WORT
INT (целое число)	27 -25		#TYP_INT
S5TIME (время в формате S5)	S5T#10MS		#TYP_S5_TIME
DATE (дата в формате IEC)	D#1995-12-24		#TYP_DATE
DWORD (двойное слово)	DW#16#FFFF_0F02 2#0001_1111_0001_0010_0001_1111_0001_0010 B#(5,4,59,8)	MD 100 ED 0 AD 0 DBD 4	#TYP_DWORT
DINT (двойное целое число)	L#170 L#-350		#TYP_DINT
REAL (вещественное число)	1.23		#TYP_REAL
TIME (время в формате IEC)	T#20MS		#TYP_TIME
TIME_OF_DAY (время суток в формате IEC)	TOD#23:59:12.3		#TYP_TOD

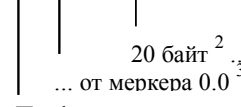
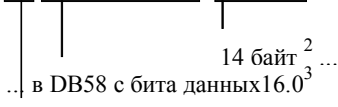
Формальный параметр типа ...	Пример назначения фактического параметра		
	Прямой ввод (значение)	Ввод элемента глобальных данных	Символический ввод ¹
DATE_AND_TIME (дата и время IEC)	невозможен ? переменная должна быть описана, например, как временная)		#TYP_8_BYTE
ANY (тип данных "любого" типа и "любой" величины)	P#M0.0 BYTE20  Префикс для указателя ANY P#DB58.DBX16.0 BYTE14  Префикс для указателя ANY	E 0.0 MB 5 AW 2 (возможно применение любого глобального операнда STEP 7)	#TYP_ANYTYP (Описание массивов и структур)

Рис. 23-1. Назначение фактических параметров, продолжение

- ¹ Предпосылка: Для глобальных данных Вы определить имя (= символ) в таблице глобальных символов, прежде чем Вы сможете использовать этот символ в качестве фактического параметра. Для локальных данных Вы должны имя (= символ) определить в таблице описаний блока, прежде чем Вы сможете использовать этот символ в качестве фактического параметра.
Для локальных данных Вы должны перед символом поставить знак #.
- ² В качестве длины можно указывать элементарные типы данных, напр., BOOL, BYTE, WORD или DWORD или составные типы данных, напр., DATE_AND_TIME.
- ³ Всегда указывать битовый адрес; при задании длины всегда вводить битовый адрес 0 (исключение: BOOL).

Условный вызов в AWL Чтобы произвести условный вызов SFC на языке программирования AWL, Вы можете использовать, например, следующую последовательность команд:

AWL	Объяснение
U #OK_MERKER SPBNB m001 CALL SFC 28 OB_NR := 10 SDT := #OUT_UHRZEIT_DATUM PERIOD := W#16#1201 RET_VAL := MW 200	Условие для вызова. Если условие не выполнено (VKE=0), вызов SFC пропускается, а VKE сохраняется в бите состояния BIE.
m001: U BIE = M 202.3	Опрос бита состояния BIE

23.4. Функции Master Control Relay

Описание

Master Control Relay (главное управляющее реле, MCR) используется в релейно-контактных схемах для активизации и деактивизации потока сигнала (пути тока). Деактивированный путь тока соответствует последовательности операций, которая записывает нулевое значение вместо рассчитанного, или последовательности операций, которая оставляет неизменным существующее значение памяти. От MCR зависят следующие логические операции и операции передачи:

- =
- S
- R
- T (применяется с байтом, словом или двойным словом)

Операция T, применяемая с байтом, словом или двойным словом, и операция = записывают в память "0", если MCR равно "0". Операции S и R не изменяют существующее значение.

Таблица 23–4. Операции, зависящие от MCR, и их реакция на сигнальное состояние MCR

Сигнальное состояние MCR	=	S или R	T
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 компоненту, которая при исчезновении напряжения передает значение "0")
1	Нормальная обработка	Нормальная обработка	Нормальная обработка

23.5. Операции для Master Control Relay

Обзор

С помощью следующих операций можно управлять Master Control Relay:

- MCRA Активизировать область MCR
- MCRD Деактивизировать область MCR
- MCR(Сохранить VKE в стеке MCR, начало области MCR
-)MCR Извлечь VKE из стека MCR, конец области MCR

Описание: MCRA, MCRD 23–3):

Следующие операции активизируют и деактивируют область MCR, т.е. они указывают, какие операции в вашей программе зависят от MCR (см. также рис.

- Активизировать область MCR: MCRA
- Деактивизировать область MCR: MCRD

Операции, запрограммированные между MCRA и MCRD, зависят от сигнального состояния бита MCR. Операции, запрограммированные вне последовательности MCRA–MCRD, не зависят от сигнального состояния бита MCR. Если операция MCRD отсутствует, то от бита MCR зависят операции, запрограммированные между MCRA и BEA (см. рис. 23–2).

Зависимость функций (FC) и функциональных блоков (FB) от MCR в блоках Вы должны программировать сами. Если эта функция или этот функциональный блок вызывается из последовательности MCRA–MCRD, то не все команды внутри этой последовательности автоматически зависят от бита MCR. Чтобы сделать операции в вызванном блоке зависящими от бита MCR, Вы должны применить операцию MCRA вызванного блока.



Опасность

Никогда не применяйте операцию MCR для аварийного отключения или предохранительное устройство для людей.

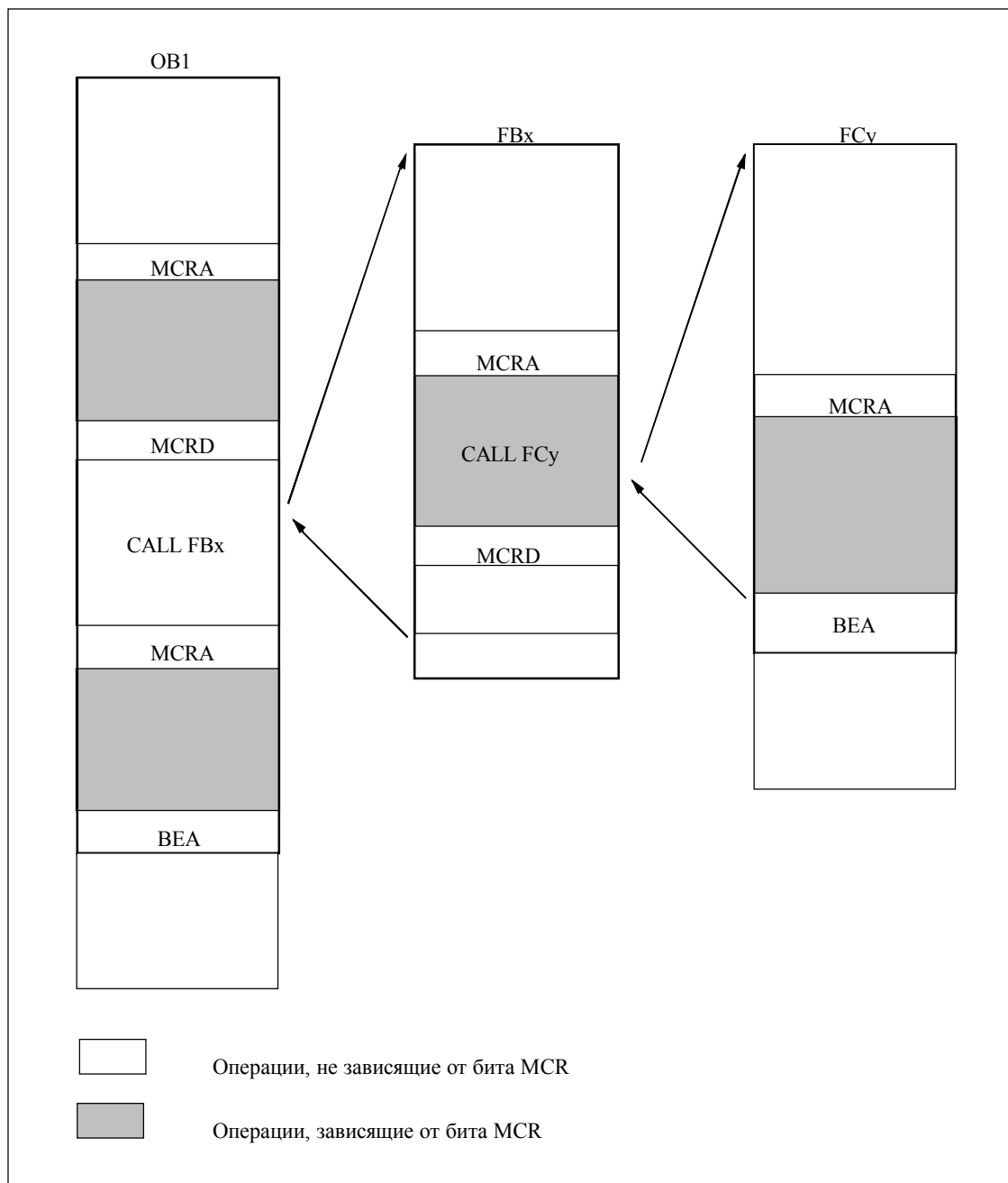


Рис. 23-2. Активизация и деактивирование области MCR

Описание:
MCR(,)MCR

Следующие операции включают или выключают функционирование Master Control Relay:

- Сохранить VKE в стеке MCR, начать область MCR: MCR(
- Извлечь VKE из стека MCR, завершить область MCR:)MCR

Операции MCR(и)MCR можно вкладывать друг в друга. Максимальная глубина вложения равна восьми, т.е. Вы можете записать друг за другом не более восьми операций MCR(, прежде чем вставите операцию)MCR. Вы должны запрограммировать одинаковое количество операций MCR(и)MCR (см. рис. 23–4).

Если операции MCR(вкладываются друг в друга, то бит MCR более глубокого уровня вложения образуется логическим соединением текущего VKE операции MCR(с текущим битом MCR в соответствии с таблицей истинности для И. Если вложение операции MCR(отсутствует, то бит MCR образуется копированием текущего VKE операции MCR в бит MCR.

Когда операция)MCR завершает уровень вложения, она извлекает бит MCR из более высокого уровня. Операция)MCR самого высокого уровня устанавливает бит MCR в “1”.

Операции MCR(и)MCR в своей программе Вы всегда должны применять парами.

Пример

На рис. 23–3 показано, как применяется Master Control Relay.

Контакт MCR замыкается, когда бит MCR равен “1”. Сигнальные состояния выходов А 4.0 и А 4.1 вычисляются в соответствии с сигнальными состояниями входов от Е 1.0 до Е 1.3 и их логическими соединениями.

Контакт MCR размыкается, когда бит MCR равен “0”. Выходы А 4.0 и А 4.1 сбрасываются в “0” независимо от сигнальных состояний входов от Е 1.0 до Е 1.3.

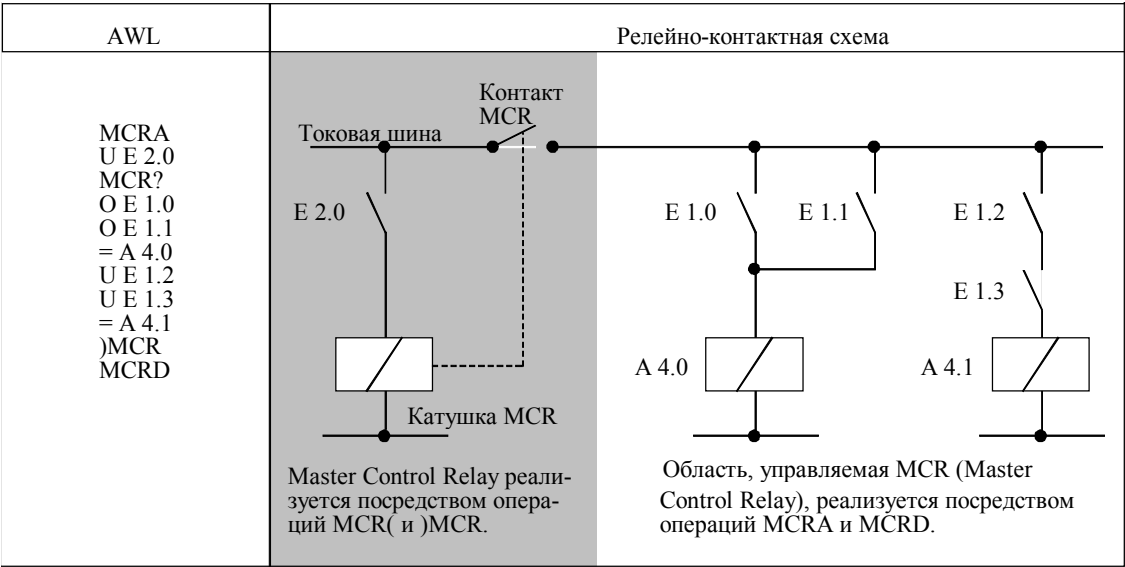
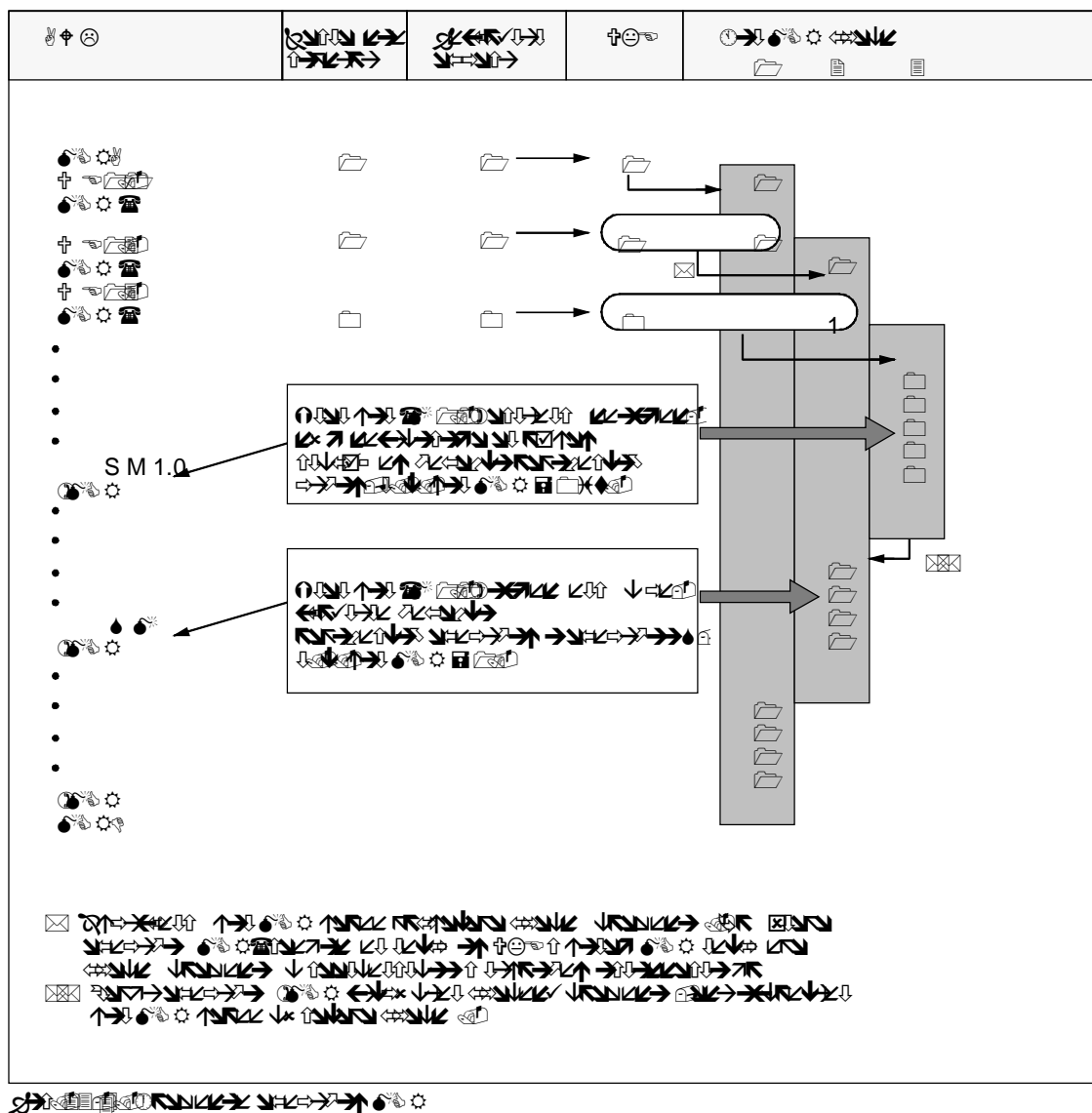


Рис. 23-3. Реализация Master Control Relay

На рис. 23–4 показано, как можно производить вложение операций.



23.6. Конец блока

Описание

Операция “Конец блока” - это команда программирования, завершающая обработку блока. Блок можно завершить одной из следующих двух операций:

- Абсолютный конец блока (BEA): Эта операция завершает обработку текущего блока и передает управление обратно в блок, который вызвал только что закончившийся блок. Когда программа сталкивается с операцией BEA, она завершает текущий блок независимо от результата логической операции.
- Условный конец блока (BEB): Эта операция завершает обработку текущего блока и передает управление обратно в блок, который вызвал только что закончившийся блок. Когда программа сталкивается с операцией BEB, она завершает текущий блок только тогда, когда результат логической операции равен "1" ($VKE = 1$). Если VKE равен "0", то программа не выполняет операцию “Конец блока”. VKE устанавливается в "1" и продолжается обработка программы в текущем блоке.