

S

SIMATIC

LAD для S7-300 и S7-400, Программирование

Справочное руководство

Это руководство является частью пакета документации с заказным номером:

6ES7810-4CA05-8BR0

Важные замечания,
содержание

Обзор продукта	1
Структура и элементы контактного плана	2
Адресация	3
Битовые логические операции	4
Таймерные команды	5
Операции со счетчиками	6
Операции с целыми числами	7
Операции над числами с плавающей точкой	8
Команды сравнения	9
Команды пересылки и преобразования	10
Поразрядные логические операции над словами	11
Команды сдвига и циклического сдвига	12
Операции с блоками данных	13
Команды перехода	14
Операции с битами состояния	15
Команды управления программой	16
Приложения	
Алфавитный список команд	A
Примеры программирования	B
Литература	C
Глоссарий	

C79000-G7000-C562-01

Указания по безопасности

Это руководство содержит указания, которые вы должны соблюдать для обеспечения собственной безопасности, а также защиты продукта и подключенного оборудования. Эти указания выделены в руководстве предупреждающим треугольником и помечены следующим образом в соответствии с уровнем опасности:



Опасность

Указывает, что несоблюдение надлежащих предосторожностей приведет к смерти, тяжким телесным повреждениям или существенному повреждению имущества.



Предупреждение

Указывает, что несоблюдение надлежащих предосторожностей может привести к смерти, тяжким телесным повреждениям или существенному повреждению имущества.



Предостережение

Указывает, что несоблюдение надлежащих предосторожностей может привести к небольшим телесным повреждениям или порче имущества.

Замечание

Привлекает ваше внимание к особенно важной информации о продукте, обращении с продуктом или к определенной части документации.

Квалифицированный персонал

К установке и работе на данном оборудовании должен допускаться только квалифицированный персонал. К квалифицированному персоналу относятся лица, имеющие право пускать в эксплуатацию, заземлять и маркировать электрические цепи, оборудование и системы в соответствии с установленным порядком и стандартами.

Правильное использование

Примите во внимание следующее:



Предупреждение

Это устройство и его компоненты могут быть использованы только для приложений, описанных в каталоге или технических описаниях, и только в соединении с устройствами или компонентами других производителей, которые были одобрены или рекомендованы фирмой Siemens.

Этот продукт может правильно и безопасно функционировать только при правильной транспортировке, хранении, установке и установке, а также эксплуатации и обслуживании в соответствии с рекомендациями.

Торговые марки

SIMATIC®, SIMATIC HMI® и SIMATIC NET® являются зарегистрированными торговыми марками SIEMENS AG.

Некоторые из других обозначений, использованных в этих документах, также являются зарегистрированными торговыми марками; права собственности могут быть нарушены, если эти обозначения используются третьей стороной для своих собственных целей.

Copyright © Siemens AG 1998 Все права сохраняются

Воспроизведение, передача или использование этого документа или его содержания не допускается без специального письменного разрешения. Нарушители будут нести ответственность за нанесенный ущерб. Все права, включая права, создаваемые патентным грантом или регистрацией сервисной модели или проекта, сохраняются.

Siemens AG
Департамент техники автоматизации и приводов
Сфера деятельности: промышленные системы автоматизации
п/я 4848, D- 90327 Нюрнберг

Отказ от ответственности

Мы проверили содержание этого руководства на соответствие с описанной аппаратурой и программным обеспечением. Так как отклонения не могут быть полностью предотвращены, мы не гарантируем полного соответствия. Однако данные, приведенные в этом руководстве, регулярно пересматриваются и необходимые исправления вносятся в последующие издания. Приветствуются предложения по улучшению.

©Siemens AG 1998
Технические данные могут изменяться.

Предисловие

Назначение

Это руководство является вашим путеводителем при создании программ пользователя на языке программирования, который в документации на русском языке часто называется *Контактный план* (в переводе с принятого в SIMATIC немецкого термина *Kontaktplan* (KOP)). Его международное название – Ladder Logic (LAD), т.е. цепная логическая схема. В дальнейшем будет обычно использоваться термин *Контактный план* и сокращенное название *KOP*.

Это руководство включает также справочный раздел, описывающий синтаксис и функции элементов контактного плана.

Круг читателей

Это руководство предназначено для программистов, операторов и обслуживающего персонала систем S7. Существенно важным является знакомство с процедурами автоматизации.

Область применения

Это руководство действительно для версии 5.0 пакета программного обеспечения STEP 7.

Соответствие стандартам

KOP соответствует языку «Ladder Logic», определенному в стандарте Международной электротехнической комиссии IEC 1131–3. Дополнительные подробности вы найдете в таблице стандартов в файле NORM_TBL.WRI пакета STEP 7.

Требования

Для эффективного использования данного руководства вы должны быть уже знакомы с теорией, на которую опирается программирование для S7 и которая задокументирована в оперативной помощи для STEP 7.

Языковые пакеты используют также стандартное программное обеспечение STEP 7, так что вы должны быть знакомы с тем, как обращаться с этим программным обеспечением, и прочитать сопроводительную документацию.

Документация	Назначение	Номер для заказа
Базовая информация о STEP 7, включающая в себя: <ul style="list-style-type: none"> Working with STEP 7 V5.0, Getting Started Manual [Работа со STEP 7 версии 5.0. Введение в STEP 7] Programming with STEP 7 V5.0 [Программирование с помощью STEP 7 версии 5.0] Configuring Hardware and Communication Connections, STEP 7 V5.0 [Конфигурирование аппаратуры и проектирование соединений с помощью STEP 7 v5.0] From S5 to S7, Converter Manual [От S5 к S7. Руководство по конвертированию] 	Базовая информация для технического персонала, описывающая методы реализации задач управления с помощью STEP 7 и программируемых контроллеров S7-300/400.	6ES7810-4CA04-8BA0
Справочники по STEP 7, в том числе <ul style="list-style-type: none"> Руководства Ladder Logic (LAD) /Function Block Diagram (FBD) /Statement List (STL) for S7-300/400 [Контактный план (LAD, KOP) /Функциональный план (FBD, FUP) /Список операторов (STL, AWL) для S7-300/400] Standard and System Functions for S7-300/400 [Стандартные и системные функции для S7-300/400] 	Предоставляется справочная информация и описываются языки программирования LAD (контактный план, KOP), FBD (функциональный план, FUP) и STL (список операторов, AWL) и стандартные и системные функции, расширяя объем базовой информации о STEP 7.	6ES7810-4CA04-8BR0

Оперативные справки	Назначение	Номер для заказа
Помощь по STEP 7	Базовая информация о программировании и конфигурировании аппаратуры с помощью STEP 7 в виде оперативной справки (online).	Часть стандартного программного обеспечения STEP 7.
Справочная информация о STL/LAD/FBD Справочная информация о SFB/SFC Справочная информация об организационных блоках	Контекстно-чувствительная справочная информация	Часть стандартного программного обеспечения STEP 7.

Доступ к оперативной помощи

Оперативную помощь вы можете отобразить следующими способами:

- Контекстно-чувствительная помощь о выделенном объекте с помощью команды меню **Help > Context-Sensitive Help [Помощь > Контекстно-чувствительная помощь]**, с помощью **функциональной клавиши F1** или щелчком на символе вопросительного знака на панели инструментов.
- Помощь по STEP 7 через команду меню **Help > Contents [Помощь > Содержание]**.

Ссылки

Ссылки на другую документацию даются с помощью номеров, заключенных между косыми чертами /.../. Используя эти номера вы можете проверить точное название документа в разделе Литература в конце данного руководства.

Оперативные службы поддержки клиентов SIMATIC

Бригада поддержки клиентов SIMATIC предлагает существенную дополнительную информацию о продуктах SIMATIC через свои оперативные службы:

- Общая текущая информация может быть получена:
 - в **Internet** под
http://www.ad.siemens.de/simatic/html_00/simatic
 - через **Fax-Polling** номер 08765–93 02 77 95 00
- Текущие данные о продукте и загрузки, которые вы, возможно, найдете полезными, доступны:
 - в **Internet** через http://www.ad.siemens.de/support/html_00/
 - через **Bulletin Board System** (BBS) в Нюрнберге (*SIMATIC Customer Support Mailbox*) под номером +49 (911) 895–7100.

Для набора почтового ящика используйте модем с протоколом до V.34 (28.8 кБод) со следующей настройкой параметров: 8, N, 1, ANSI; или наберите через ISDN (x.75, 64 кБод).

Дополнительная помощь

Если у вас есть другие вопросы, обращайтесь, пожалуйста, к представителю фирмы Siemens в вашем регионе. Адреса перечислены, например, в каталогах и в CompuServe (*go autforum*).

Наша горячая линия **SIMATIC Basic Hotline** тоже готова помочь:

- в Нюрнберге, Германия
 - с понедельника по пятницу с 07:00 до 17:00 (местное время):
телефон: +49 (911) 895-7000
 - или E-mail: simatic.support@nbgm.siemens.de
- в Джонсон-Сити (TN), USA
 - с понедельника по пятницу с 08:00 до 17:00 (местное время):
телефон: +1 423 461-2522
 - или E-mail: simatic.hotline@sea.siemens.com
- в Сингапуре
 - с понедельника по пятницу с 08:30 до 17:30 (местное время):
телефон: +65 740-7000
 - или E-mail: simatic@singet.com.sg

Платная горячая линия SIMATIC Premium Hotline доступна круглосуточно по всему миру с помощью SIMATIC card (телефон: +49 (911) 895–7777).

Курсы по продуктам SIMATIC

Фирма Siemens предлагает ряд учебных курсов для ознакомления с системой автоматизации SIMATIC S7. Для получения более подробной информации обращайтесь в свой региональный учебный центр или в центральный учебный центр Нюрнберге, Германия:
Телефон: +49 (911) 895–3154.

Содержание

Предисловие

1	Обзор продукта	1–1
2	Структура и элементы контактного плана	2–1
2.1	Элементы и блоки	2–2
2.2	Булева логика и таблицы истинности	2–6
2.3	Значение регистров CPU в командах	2–12
3	Адресация	3–1
3.1	Обзор	3–2
3.2	Виды адресов	3–4
4	Битовые логические операции	4–1
4.1	Обзор	4–2
4.2	Нормально открытый контакт	4–3
4.3	Нормально замкнутый контакт	4–4
4.4	Выходная катушка	4–5
4.5	Промежуточный выход (коннектор)	4–6
4.6	Инвертирование результата логической операции	4–7
4.7	Сохранение RLO в регистре BR	4–8
4.8	Установка выхода	4–9
4.9	Сброс выхода	4–10
4.10	Установка начального значения счетчика	4–11
4.11	Катушка со счетчиком прямого счета	4–12
4.12	Катушка со счетчиком обратного счета	4–13
4.13	Катушка с таймером – формирователем импульса	4–14
4.14	Катушка с таймером – формирователем удлиненного импульса	4–15
4.15	Катушка с таймером – формирователем задержки включения	4–16
4.16	Катушка с таймером – формирователем задержки включения с запоминанием	4–17
4.17	Катушка с таймером – формирователем задержки выключения	4–18
4.18	Обнаружение положительного фронта RLO	4–19
4.19	Обнаружение отрицательного фронта RLO	4–20
4.20	Обнаружение положительного фронта сигнала	4–21
4.21	Обнаружение отрицательного фронта сигнала	4–22
4.22	Установка-сброс триггера	4–23
4.23	Сброс-установка триггера	4–24
5	Таймерные команды	5–1
5.1	Размещение таймера в памяти и его компоненты	5–2
5.2	Выбор подходящего таймера	5–4
5.3	Таймер S5 – формирователь импульса	5–5
5.4	Таймер S5 – формирователь удлиненного импульса	5–8

5.5	Таймер S5 – формирователь задержки включения	5–11
5.6	Таймер S5 – формирователь задержки включения с запоминанием	5–14
5.7	Таймер S5 – формирователь задержки выключения	5–17
6	Операции со счетчиками	6–1
6.1	Размещение счетчика в памяти и его компоненты	6–2
6.2	Счетчик прямого и обратного счета	6–4
6.3	Счетчик прямого счета	6–6
6.4	Счетчик обратного счета	6–8
7	Операции с целыми числами	7–1
7.1	Сложение целых чисел	7–2
7.2	Сложение двойных целых чисел	7–3
7.3	Вычитание целых чисел	7–4
7.4	Вычитание двойных целых чисел	7–5
7.5	Умножение целых чисел	7–6
7.6	Умножение двойных целых чисел	7–7
7.7	Деление целых чисел	7–8
7.8	Деление двойных целых чисел	7–9
7.9	Получение остатка от деления двойного целого числа	7–10
7.10	Оценка битов слова состояния после операций с целыми числами	7–11
8	Операции над числами с плавающей точкой	8–1
8.1	Обзор	8–2
8.2	Сложение чисел с плавающей точкой	8–3
8.3	Вычитание чисел с плавающей точкой	8–4
8.4	Умножение чисел с плавающей точкой	8–5
8.5	Деление чисел с плавающей точкой	8–6
8.6	Оценка битов слова состояния после выполнения операций с плавающей точкой	8–7
8.7	Образование абсолютного значения числа с плавающей точкой	8–8
8.8	Образование квадрата и/или квадратного корня числа с плавающей точкой	8–9
8.9	Образование натурального логарифма числа с плавающей точкой	8–11
8.10	Образование экспоненциального значения числа с плавающей точкой	8–12
8.11	Образование тригонометрических функций углов в виде чисел с плавающей точкой	8–13
9	Команды сравнения	9–1
9.1	Сравнение целых чисел	9–2
9.2	Сравнение двойных целых чисел	9–3
9.3	Сравнение чисел с плавающей точкой	9–5

10	Команды пересылки и преобразования	10–1
10.1	Присваивание значения	10–2
10.2	Преобразование двоично-десятичного числа в целое	10–4
10.3	Преобразование целого числа в двоично-десятичное	10–5
10.4	Преобразование целого числа в двойное целое	10–6
10.5	Преобразование двоично-десятичного числа в двойное целое	10–7
10.6	Преобразование двойного целого числа в двоично-десятичное	10–8
10.7	Преобразование двойного целого числа в число с плавающей точкой	10–9
10.8	Дополнение целого числа до единицы	10–10
10.9	Дополнение двойного целого числа до единицы	10–11
10.10	Дополнение целого числа до двух	10–12
10.11	Дополнение двойного целого числа до двух	10–13
10.12	Изменение знака числа с плавающей точкой	10–14
10.13	Округление до двойного целого числа	10–15
10.14	Выделение целой части числа	10–16
10.15	Округление до ближайшего большего целого числа	10–17
10.16	Округление до ближайшего меньшего целого числа	10–18
11	Поразрядные логические операции над словами	11–1
11.1	Обзор	11–2
11.2	Поразрядное И над словами	11–3
11.3	Поразрядное И над двойными словами	11–5
11.4	Поразрядное ИЛИ над словами	11–7
11.5	Поразрядное ИЛИ над двойными словами	11–9
11.6	Поразрядное исключающее ИЛИ над словами	11–11
11.7	Поразрядное исключающее ИЛИ над двойными словами	11–13
12	Команды сдвига и циклического сдвига	12–1
12.1	Команды сдвига	12–2
12.2	Команды циклического сдвига	12–10
13	Операции с блоками данных	17–1
13.1	Открытие блока данных: DB или DI	17–2
14	Команды перехода	14–1
14.1	Обзор	14–2
14.2	Переход в блоке, если RLO = 1 (безусловный переход)	14–3
14.3	Переход в блоке, если RLO = 1 (условный переход)	14–4
14.4	Переход в блоке, если RLO = 0 (переход, если не 1)	14–5
14.5	Метка	14–6
15	Операции с битами состояния	15–1
15.1	Обзор	15–2
15.2	Бит ошибки "Регистр BR"	15–3
15.3	Биты результата	15–4
15.4	Бит ошибки "Недопустимая операция"	15–6
15.5	Бит ошибки "Переполнение"	15–7
15.6	Бит ошибки "Сохраняемое переполнение"	15–8

16	Команды управления программой	16–1
16.1	Вызов FC/SFC из катушки	16–2
16.2	Вызов FB, FC, SFB, SFC и мультитекстов	16–4
16.3	Возврат	16–7
16.4	Команды главного управляющего реле	16–9
16.5	Активизация/деактивизация главного управляющего реле	16–10
16.6	Включение/выключение главного управляющего реле	16–13
Приложения		
A	Алфавитный список команд	A–1
A.1	Список международных наименований	A–2
A.2	Список международных наименований с их русскими соответствиями	A–5
A.3	Список русских наименований	A–8
A.4	Список русских наименований и их международных соответствий	A–11
A.5	Список международных сокращенных наименований и сокращенных наименований SIMATIC	A–14
B	Примеры программирования	B–1
B.1	Обзор	B–2
B.2	Битовые логические операции	B–3
B.3	Таймерные команды	B–7
B.4	Операции счета и сравнения	B–11
B.5	Арифметические операции с целыми числами	B–13
B.6	Логические операции со словами	B–14
C	Литература	C–1
Глоссарий		Глоссарий–1

1 Обзор продукта

Что такое KOP (LAD)?

KOP – это сокращенное обозначение контактного плана (от немецкого Kontaktplan). LAD – это сокращение международного (английского) термина Ladder Logic [цепная логическая схема]. KOP - это графический язык программирования. Синтаксис команд похож на коммутационную схему. KOP позволяет легко проследить поток сигнала между токовыми шинами через входы, выходы и команды.

Язык программирования Контактный план

Язык программирования Контактный план имеет все необходимые элементы для создания полной программы пользователя. Он содержит полный набор базовых команд, и в вашем распоряжении имеется широкий диапазон адресов. Функции и функциональные блоки позволяют наглядно структурировать вашу программу на KOP.

Пакет программ

Пакет программ KOP - встроенная составная часть стандартного программного обеспечения STEP 7. Это значит, что после установки вами программного обеспечения STEP 7 в вашем распоряжении имеются все функции редактирования, компиляции и тестирования/отладки для KOP.

Используя KOP, вы можете создавать свои прикладные программы в инкрементном редакторе. В нем удобно решен ввод локальных для блока структур данных с помощью редакторов таблиц.

В стандартном программном обеспечении имеется три языка программирования – STL (AWL), FBD (FUP) и LAD (KOP). Вы можете почти без ограничений переходить с одного языка на другой и выбирать наиболее подходящий язык для конкретного блока, который вы программируете.

Если вы пишете программы на LAD (KOP) или FBD (FUP), то вы всегда можете переключиться на представление STL (AWL). Если вы преобразуете программы на KOP в программы на FUP и наоборот, то элементы программы, которые не могут быть представлены в целевом языке, отображаются на AWL.

2 Структура и элементы контактного плана

Обзор главы

Раздел	Описание	Стр.
2.1	Элементы и блоки	2–2
2.2	Булева логика и таблицы истинности	2–6
2.3	Значение регистров CPU в командах	2–12


2.1 Элементы и блоки

Команды KOP

Команды KOP состоят из элементов и блоков, графически объединяемых в сегменты. Элементы и блоки можно разделить на следующие группы:

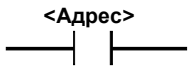
Команды как элементы

STEP 7 представляет некоторые команды контактного плана в виде отдельных элементов, которым не нужны ни адреса, ни параметры (см. таблицу 2–1).

Таблица 2–1. Команда контактного плана как элемент без адреса и параметров		
Элемент	Название	Раздел в данном руководстве
	Инvertировать результат логической операции (поток энергии)	4.6

Команды как элементы с адресом

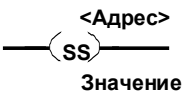
STEP 7 представляет некоторые команды контактного плана как отдельные элементы, для которых нужно вводить адрес (см. таблицу 2–2). Для получения дополнительной информации об адресации см. главу 3.

Таблица 2–2. Команда контактного плана как элемент с адресом		
Элемент	Название	Раздел в данном руководстве
	Нормально открытый контакт	4.2

Команды как элементы с адресом и значением

STEP 7 представляет некоторые команды контактного плана как отдельные элементы, для которых нужно вводить адрес и значение (например, время или счетное значение, см. таблицу 2–3).

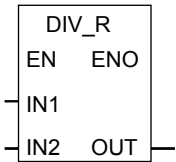
Для получения дополнительной информации об адресации см. главу 3.

Таблица 2–3. Команда контактного плана как элемент с адресом и значением		
Элемент	Название	Раздел в данном руководстве
	Катушка с таймером – формирователем задержки включения с запоминанием	4.16

Команды в виде блоков с параметрами

STEP 7 представляет некоторые команды контактного плана в виде блоков с линиями входов и выходов (см. таблицу 2–4). Входы находятся с левой стороны блока; выходы – с правой стороны блока. Вы заполняете входные параметры. Для выходных параметров вы указываете места, куда программное обеспечение STEP 7 может поместить для вас выходную информацию. Для этих параметров вы должны использовать нотацию, соответствующую отдельным типам данных.

Как действуют параметры EN (разрешить вход) и ENO (разрешить выход) объяснено ниже. Для получения дополнительной информации о входных и выходных параметрах обратитесь к описанию каждой команды в данном руководстве.

Таблица 2–4. Команда контактного плана в виде блока с входами и выходами		
Блок	Название	Раздел в данном руководстве
	Деление вещественных чисел	8.5

Параметры "Разрешить вход" и "Разрешить выход"

Передача энергии на разрешающий вход (EN) блока контактного плана (активизация) приводит к тому, что блок выполняет заданную функцию. Если блок способен выполнить свою функцию без ошибок, то разрешающий выход (ENO) передает энергию по цепи. Параметры блока KOP EN и ENO имеют тип BOOL и могут находиться в области памяти I, Q, M, D или L (см. таблицы 2–5 и 2–6).

EN и ENO функционируют в соответствии со следующими принципами:

- Если EN не активизирован (т.е. состояние сигнала равно 0), то блок не выполняет свою функцию, и ENO не активизируется (т.е. состояние сигнала тоже равно 0).
- Если EN активизирован (т.е. состояние сигнала равно 1) и соответствующий блок выполняет свою функцию без ошибок, то ENO тоже активизируется (т.е. состояние сигнала тоже равно 1).
- Если EN активизирован (т.е. состояние сигнала равно 1) и при обработке функции возникает ошибка, то ENO не активизируется (т.е. состояние сигнала равно 0).

Ограничения для блоков и катушек

Вы не можете размещать блоки и коннекторы в цепи контактного плана, которая не начинается на левой питающей шине. Исключением являются операции сравнения.

Области памяти и их функции

Большинство адресов в KOP относятся к областям памяти. Следующая таблица показывает их типы и функции.

Таблица 2–5. Области памяти и их функции

Название области	Функция области	Доступ к области	
		через единицы следующего размера:	Сокращение
Вход образа процесса	В начале цикла сканирования операционная система входы из процесса и записывает полученные значения в эту область. Программа может использовать эти значения при циклической обработке.	Входной бит Входной байт Входное слово Входное двойное слово	I IB IW ID
Выход образа процесса	Во время цикла сканирования программа рассчитывает выходные значения и помещает их в эту область. В конце цикла сканирования операционная система считывает рассчитанные выходные значения из этой области и передает их на выходы процесса.	Выходной бит Выходной байт Выходное слово Выходное двойное слово	Q QB QW QD
Битовая память (меркеры)	Эта область предоставляет место для хранения промежуточных результатов расчетов, выполненных в программе.	Бит памяти (меркер) Байт памяти (меркерный байт) Слово памяти (меркерное слово) Двойное слово памяти (двойное меркерное слово)	M MB MW MD
Периферийная область: внешний вход	Эта область позволяет вашей программе непосредственно обращаться к модулям ввода и вывода (т.е. к периферийным входам и выходам).	Периферийный входной байт Периферийное входное слово Периферийное входное двойное слово	PIB PIW PID
Периферийная область: внешний выход		Периферийный выходной байт Периферийное выходное слово Периферийное выходное двойное слово	PQB PQW PQD
Таймеры	Эта область предоставляет место в памяти для таймерных ячеек. В этой области датчик импульсов времени обращается к таймерным ячейкам для их актуализации путем уменьшения значения времени. Таймерные операции обращаются к этим ячейкам.	Таймер (Т)	T
Счетчики	Счетчики – это функциональные элементы КОР. Эта область предоставляет место в памяти для счетчиков. К ней обращаются операции счета.	Счетчик (С)	C
Блок данных	Эта область содержит данные, к которым можно обратиться из любого блока. Если вам нужно открыть одновременно два блока, то вы можете открыть один из них командой «OPN DB», а другой - командой «OPN DI». Нотация адресов, например, L DBWi или L DIWi определяет блок, к которому производится обращение. Хотя вы можете использовать команду «OPN DI» для открытия любого блока данных, она используется главным образом для открытия экземплярных блоков данных, связанных с функциональными блоками (FB) и с системными функциональными блоками (SFB). За дополнительной информацией о FB и SFB обращайтесь к оперативной помощи STEP 7.	Блок данных, открытый командой «OPN DB»: Бит данных Байт данных Слово данных Двойное слово данных Блок данных, открытый командой «OPN DI»: Бит данных Байт данных Слово данных Двойное слово данных	DBX DBB DBW DBD DIX DIB DIW DID

Таблица 2–5. Области памяти и их функции

Название области	Функция области	Доступ к области	
		через единицы следующего размера:	Сокращение
Локальные данные	Эта область содержит временные данные, используемые внутри логического блока (FB или FC). Эти данные называются динамическими локальными данными. Они служат в качестве промежуточной памяти. Когда кодовый блок закрывается, эти данные теряются. Эти данные содержатся в стеке локальных данных (L–стек).	Бит временных локальных данных	L
		Байт временных локальных данных	LB
		Слово временных локальных данных	LW
		Двойное слово временных локальных данных	LD

В таблице 2–6 перечислены максимальные диапазоны адресов для различных областей памяти. За диапазонами адресов, возможных для вашего CPU, обратитесь к соответствующему руководству для CPU S7–300.

Таблица 2–6. Области памяти и их диапазоны адресов

Название области	Доступ к области		Максимальный диапазон адресов
	через единицы следующих размеров:	Сокр.	
Вход образа процесса	Входной бит Входной байт Входное слово Входное двойное слово	I IB IW ID	от 0.0 до 65 535.7 от 0 до 65 535 от 0 до 65 534 от 0 до 65 532
Выход образа процесса	Выходной бит Выходной байт Выходное слово Выходное двойное слово	Q QB QW QD	от 0.0 до 65 535.7 от 0 до 65 535 от 0 до 65 534 от 0 до 65 532
Битовая память (меркеры)	Бит памяти (меркер) Байт памяти (меркерный байт) Слово памяти (меркерное слово) Двойное слово памяти (двойное меркерное слово)	M MB MW MD	от 0.0 до 255.7 от 0 до 255 от 0 до 254 от 0 до 252
Периферийная область: внешний вход	Периферийный входной байт	PIB	от 0 до 65 535
Периферийная область: внешний выход	Периферийное входное слово	PIW	от 0 до 65 534
	Периферийное входное двойное слово	PID	от 0 до 65 532
	Периферийный выходной байт	PQB	от 0 до 65 535
	Периферийное выходное слово	PQW	от 0 до 65 534
	Периферийное выходное двойное слово	PQD	от 0 до 65 532
Таймеры	Таймер (T)	T	от 0 до 255
Счетчики	Счетчик (C)	C	от 0 до 255
Блок данных	Блок данных, открытый командой DB --(OPN) Бит данных Байт данных Слово данных Двойное слово данных Блок данных, открытый командой DI --(OPN) Бит данных Байт данных Слово данных Двойное слово данных	 DBX DBB DBW DBD DIX DIB DIW DID	 от 0.0 до 65 535.7 от 0 до 65 535 от 0 до 65 534 от 0 до 65 532 от 0.0 до 65 535.7 от 0 до 65 535 от 0 до 65 534 от 0 до 65 532
Локальные данные	Бит временных локальных данных Байт временных локальных данных Слово временных локальных данных Двойное слово временных локальных данных	L LB LW LD	от 0.0 до 65 535.7 от 0 до 65 535 от 0 до 65 534 от 0 до 65 532

2.2 Булева логика и таблицы истинности

Поток энергии

Программа КОР следит за тем, как поток энергии проходит между питающими шинами через различные входы, выходы и другие элементы и блоки. Многие команды контактного плана действуют в соответствии с принципами булевой логики.

Каждая логическая операция опрашивает состояние сигнала электрического контакта на 0 (не активизирован, выключен) или на 1 (активизирован, включен) и вслед за этим выдает результат. Затем операция или сохраняет это результат, или использует его для выполнения булевой логической операции. Результат логической операции называется RLO (в мнемонике SIMATIC – VKE). Принципы булевой логики демонстрируются здесь с помощью замыкающих и размыкающих контактов.

Нормально открытый (замыкающий) контакт

На рис. 2-1 показаны два представления релейно-контактной схемы с одним контактом реле между питающей шиной и катушкой. В нормальном состоянии этот контакт открыт. Если контакт не активизирован, он остается открытым. Сигнальное состояние открытого контакта равно 0 (не активизирован). Если контакт остается открытым, энергия от питающей шины не может возбудить катушку в конце цепи. Если контакт активизирован (состояние сигнала контакта равно 1), то ток будет проходить через катушку.

Цепь слева на рис. 2-1 показывает нормально открытый контакт реле, как его иногда представляют на релейно-контактных схемах. В данном примере цепь справа показывает, что контакт был активизирован, и поэтому замкнут.

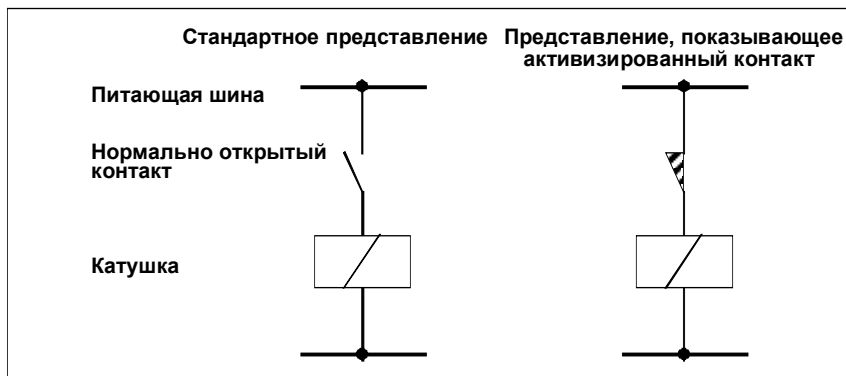


Рис. 2–1. Релейно-контактная схема с нормально открытым контактом управляющего реле

Вы можете использовать команду *Нормально открытый контакт* (см. раздел 4.2) для опроса состояния сигнала нормально открытого контакта управляющего реле. Опрашивая состояние сигнала, команда определяет, может ток протекать через контакт или нет. Если ток может протекать, то команда дает результат 1; если ток не может протекать, то команда дает результат 0 (см. таблицу 2–7). Команда может сохранить этот результат или использовать его для выполнения булевой логической операции.

Нормально замкнутый (размыкающий) контакт

На рис. 2-2 показаны два представления релейно-контактной схемы с одним контактом между питающей шиной и катушкой. В нормальном состоянии этот контакт замкнут. Если контакт не активизирован, он остается замкнутым. Сигнальное состояние замкнутого контакта равно нулю (не активизирован). Если контакт остается замкнутым, ток от питающей шины может пересечь контакт, чтобы возбудить катушку на конце цепи. Активизация контакта (сигнальное состояние контакта равно 1) открывает контакт, прерывая протекание тока через катушку.

Цепь слева на рис. 2-2 показывает нормально замкнутый контакт реле, как его иногда представляют на релейно-контактных схемах. В данном примере цепь справа показывает, что контакт был активизирован, и поэтому открыт.

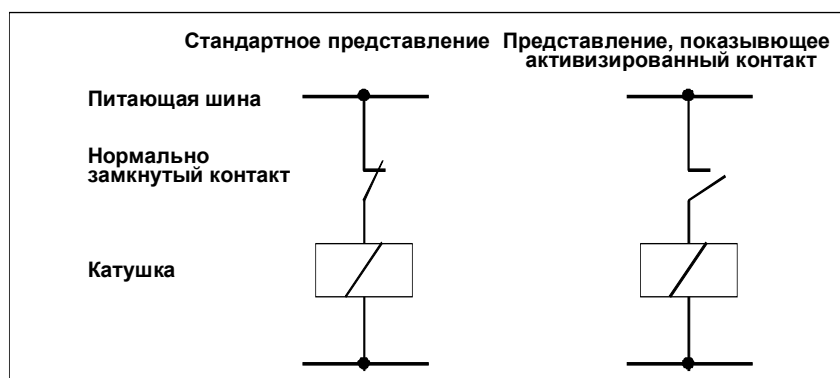


Рис. 2–2 Релейно-контактная схема с нормально замкнутым контактом управляющего реле

Вы можете использовать команду *Нормально замкнутый контакт* (см. раздел 4.3) для опроса состояния сигнала нормально замкнутого контакта управляющего реле. Опрашивая состояние сигнала, команда определяет, может ток протекать через контакт или нет. Если ток может протекать, то команда дает результат 1; если ток не может протекать, то команда дает результат 0 (см. таблицу 2–7). Команда может сохранить этот результат или использовать его для выполнения булевой логической операции.

Таблица 2–7. Результат опроса состояния сигнала нормально открытым и нормально замкнутым контактом		
Команда	Результат, если сигнальное состояние контакта равно 1 (контакт активизирован)	Результат, если сигнальное состояние контакта равно 0 (контакт не активизирован)
	1 (Имеющаяся в распоряжении энергия может передаваться, так как нормально открытый контакт замкнут.)	0 (Имеющаяся в распоряжении энергия не может передаваться, так как нормально открытый контакт открыт.)
	0 (Имеющаяся в распоряжении энергия не может передаваться, так как нормально замкнутый контакт открыт.)	1 (Имеющаяся в распоряжении энергия может передаваться, так как нормально замкнутый контакт замкнут.)

Программирование контактов, соединенных последовательно

На рис. 2-3 показана цепь логических операций КОР, представляющая два нормально открытых контакта, соединенных последовательно с катушкой. Контакты помечены символом «I», что означает «вход», а катушка помечена символом «Q», что означает «выход». Активизация нормально открытого контакта замыкает контакт. Если оба контакта в логической цепи активизированы, т.е. замкнуты, то ток течет от питающей шины через контакты, возбуждая катушку на конце цепи. То есть, когда оба контакта I 1.0 и I 1.1 активизированы, ток протекает через катушку.

На схеме 1 оба контакта активизированы. Активизация нормально открытого контакта замыкает контакт. Ток течет от питающей шины через оба замкнутых контакта, возбуждая катушку на конце цепи.

На схемах 2 и 3, так как один из двух контактов не активизирован, ток не может протекать через катушку. Катушка не возбуждена.

На схеме 4 ни один из контактов не активизирован. Оба контакта остаются открытыми. Ток не может протекать через катушку. Катушка не возбуждена.

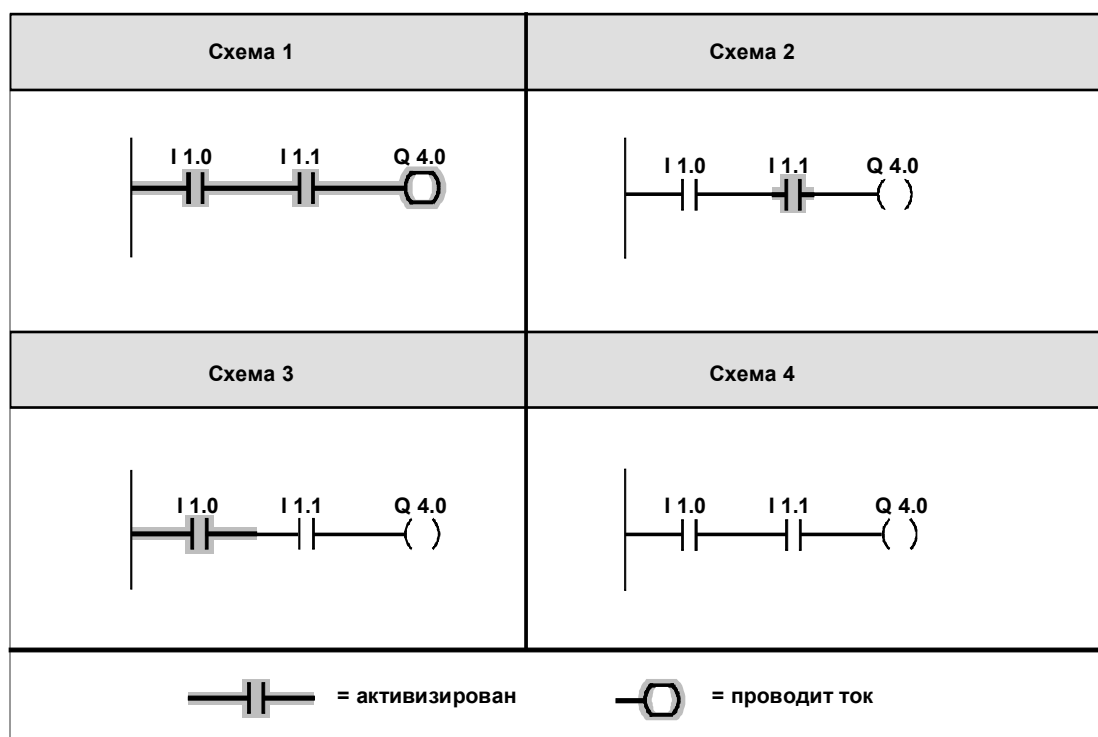


Рис. 2–3. Использование нормально открытого контакта для программирования последовательно соединенных контактов

Использование нормально открытого контакта в последовательном соединении

На рис. 2-3 показан контактный план, который вы можете использовать для программирования двух нормально открытых контактов, последовательно соединенных с катушкой реле. Первая команда *Нормально открытый контакт* в логической цепи опрашивает состояние сигнала первого контакта в последовательном соединении (вход I 1.0) и выдает основанный на этом результат (см. таблицу 2–7). Этим результатом может быть 1 или 0. Результат 1 означает, что контакт замкнут, и ток может протекать через контакт; результат 0 означает, что контакт открыт, прерывая протекание тока через контакт. Первая команда *Нормально открытый контакт* копирует эту 1 или 0 в бит памяти в слове состояния программируемого логического контроллера. Этот бит называется битом "результата логической операции" (RLO).

Вторая команда *Нормально открытый контакт* в логической цепи опрашивает сигнальное состояние второго контакта в последовательном соединении (I 1.1) и выдает основанный на этом результат (см. таблицу 2–7). Этим результатом может быть 1 или 0 в зависимости от того, замкнут контакт или открыт. В этот момент вторая команда *Нормально открытый контакт* выполняет булево логическое сопряжение. Команда берет результат, полученный ею при опросе сигнального состояния второго контакта, и логически сопрягает этот результат со значением, хранящимся в бите RLO. Результат этого сопряжения (1 или 0) сохраняется в бите RLO слова состояния, заменяя хранившееся там старое значение. Команда *Выходная катушка* (см. раздел 4.4) присваивает это новое значение катушке (выход Q 4.0).

Возможные результаты этого логического сопряжения могут быть представлены в "таблице истинности". При этом 1 означает "истину", а 0 – "ложь". Возможные логические сопряжения и их результаты обобщены в табл. 2–8. При этом слова "контакт замкнут" и "ток может протекать" соответствуют высказыванию "истина", а "контакт разомкнут" и "ток не может протекать" высказыванию "ложь" (для контактов см. рис. 2–3).

Таблица 2–8. Таблица истинности: И

Если результат опроса состояния сигнала контакта I 1.0 равен	и результат опроса состояния сигнала контакта I 1.1 равен	то результат логической операции, показанной на рис. 2–3 равен
1 (контакт замкнут)	1 (контакт замкнут)	1 (ток может протекать)
0 (контакт разомкнут)	1 (контакт замкнут)	0 (ток не может протекать)
1 (контакт замкнут)	0 (контакт разомкнут)	0 (ток не может протекать)
0 (контакт разомкнут)	0 (контакт разомкнут)	0 (ток не может протекать)

Программирование контактов, соединенных параллельно

На рис. 2-4 показана цепь логических операций КОР, представляющая два параллельно соединенных нормально открытых контакта, подключенных к катушке. Контакты помечены символом «I», что означает «вход», а катушка помечена символом «Q», что означает «выход». Активизация нормально открытого контакта замыкает контакт. Если один контакт в логической цепи (I 1.1) **или** другой контакт в этой цепи (I 1.0) активизирован (т.е. замкнут), то ток течет от питающей шины, возбуждая катушку (Q 4.0) на конце цепи. Если в логической цепи активизированы оба контакта, то ток от питающей шины может протекать, возбуждая катушку.

На схемах 1 и 2 один контакт активизирован, а другой нет. Активизация нормально открытого контакта замыкает контакт. Ток течет от питающей шины через замкнутый контакт и далее через катушку на конце цепи. Так как оба контакта соединены параллельно, то для протекания тока через катушку на конце цепи и ее возбуждения достаточно, чтобы был замкнут только один из них.

На схеме 3 оба контакта активизированы, позволяя току протекать через два замкнутых контакта до конца цепи, возбуждая катушку.

На схеме 4 ни один из контактов не активизирован. Оба контакта остаются открытыми. Ток не может протекать через катушку. Катушка не возбуждена.

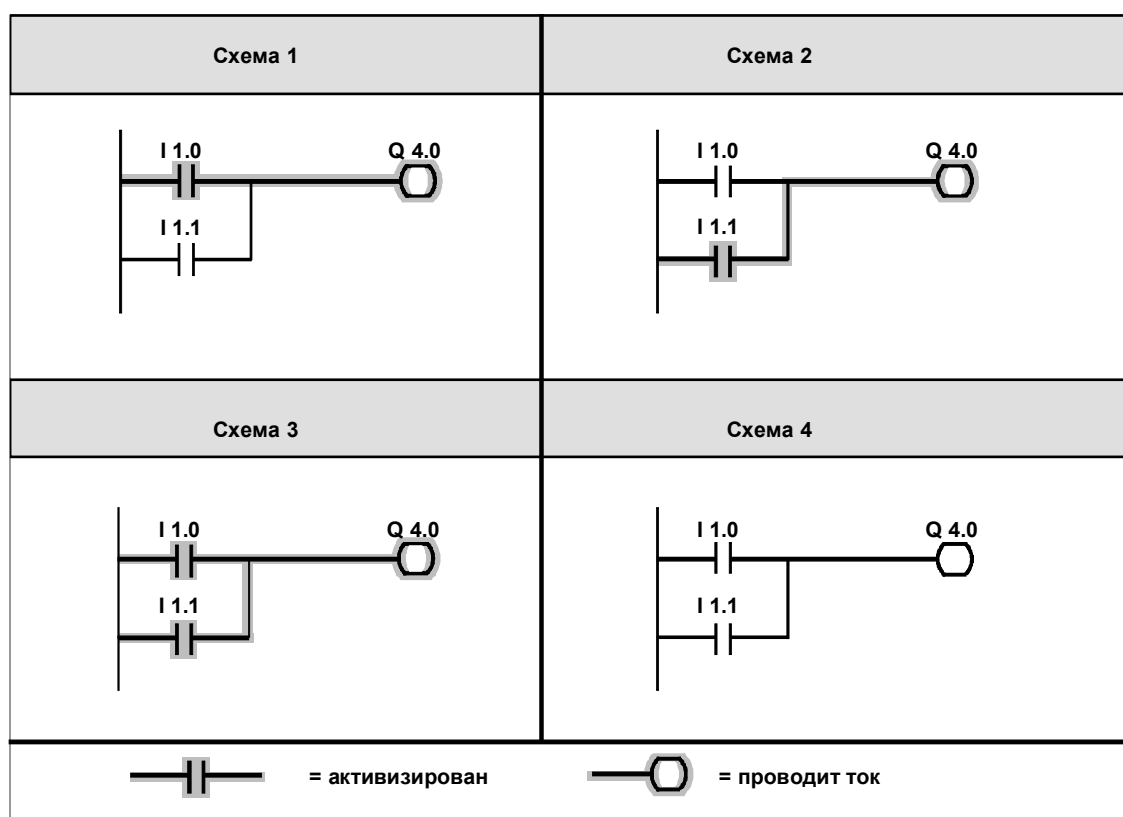


Рис. 2–4. Использование нормально открытого контакта для программирования параллельно соединенных контактов

Использование нормально открытого контакта в параллельном соединении

На рис. 2-4 показан контактный план, который вы можете использовать для программирования двух параллельно соединенных нормально открытых контактов, подключенных к катушке реле. Первая команда *Нормально открытый контакт* в логической цепи опрашивает состояние сигнала первого контакта (вход I 1.0) и выдает основанный на этом результат (см. таблицу 2–7). Этим результатом может быть 1 или 0. Результат 1 означает, что контакт замкнут, и ток может протекать через контакт; результат 0 означает, что контакт открыт, прерывая протекание тока через контакт. Первая команда *Нормально открытый контакт* копирует эту 1 или 0 в бит памяти в слове состояния программируемого логического контроллера. Этот бит называется битом "результата логической операции" (RLO).

Вторая команда *Нормально открытый контакт* в логической цепи опрашивает сигнальное состояние второго контакта (I 1.1) и выдает основанный на этом результат (см. таблицу 2–7). Этим результатом может быть 1 или 0 в зависимости от того, замкнут контакт или открыт. В этот момент вторая команда *Нормально открытый контакт* выполняет булево логическое сопряжение. Команда берет результат, полученный ею при опросе сигнального состояния второго контакта, и логически сопрягает этот результат со значением, хранящимся в бите RLO. Результат этого сопряжения (1 или 0) сохраняется в бите RLO слова состояния, заменяя хранившееся там старое значение. Команда *Выходная катушка* присваивает это новое значение катушке (выход Q 4.0).

Возможные результаты этого логического сопряжения могут быть представлены в "таблице истинности". При этом 1 означает "истину", а 0 – "ложь". Возможные логические сопряжения и их результаты обобщены в табл. 2–9. При этом слова "контакт замкнут" и "ток может протекать" соответствуют высказыванию "истина", а "контакт разомкнут" и "ток не может протекать" высказыванию "ложь" (для контактов см. рис. 2–4).

Таблица 2–9. Таблица истинности: ИЛИ		
Если результат опроса состояния сигнала контакта I 1.0 равен	и результат опроса состояния сигнала контакта I 1.1 равен	то результат логической операции, показанной на рис. 2–4 равен
1 (контакт замкнут)	0 (контакт разомкнут)	1 (ток может протекать)
0 (контакт разомкнут)	1 (контакт замкнут)	1 (ток может протекать)
1 (контакт замкнут)	1 (контакт замкнут)	1 (ток может протекать)
0 (контакт разомкнут)	0 (контакт разомкнут)	0 (ток не может протекать)

2.3 Значение регистров CPU в командах

Объяснение

Регистры помогают CPU выполнять логические и арифметические операции, операции сдвига и операции преобразования. Эти регистры описаны ниже.

Аккумуляторы

Аккумуляторы - это регистры общего назначения для обработки байтов, слов и двойных слов. Они имеют размер 32 бита.



Рис. 2–5. Области аккумулятора

Слово состояния

Слово состояния содержит биты, к которым вы можете обращаться в операндах битовых логических операций. Следующие разделы объясняют значения битов с 0 по 8.

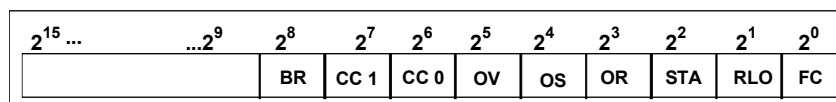


Рис. 2–6. Структура слова состояния

Изменение битов в слове состояния

Значение	Объяснение
0	Устанавливает состояние сигнала в 0
1	Устанавливает состояние сигнала в 1
x	Изменяет состояние
-	Состояние остается неизменным

Первичный опрос

Бит 0 слова состояния называется битом первичного опроса (бит FC, см. рис. 2–6). В начале сегмента KOP состояние сигнала бита FC всегда равно 0, если только предыдущий сегмент не закончился командой --(SAVE). (Черта над FC указывает, что он инвертирован, т.е. состояние его сигнала в начале сегмента контактного плана всегда равно 0).

Каждая логическая команда опрашивает состояние сигнала бита FC и контакта, к которому адресуется эта команда. Сигнальное состояние бита /ER определяет последовательность выполнения логической цепочки. Если бит FC равен 0 (в начале сегмента KOP), то команда сохраняет результат в бите результата логической операции слова состояния и устанавливает бит FC в 1. Этот процесс называется первичным опросом. К 1 или 0, сохраняемым после первичного опроса в бите RLO, затем обращаются как к результату первичного опроса.

Если сигнальное состояние бита FC равно 1, то операция логически сопрягает результат опроса состояния сигнала на обрабатываемом ею контакте с ранее образованным RLO и сохраняет этот результат в бите RLO.

Последовательность команд контактного плана (цепь логических операций) всегда заканчивается командой вывода (“Установить выход”, “Сбросить выход”, “Выходная катушка”) или командой перехода, связанной с результатом логической операции. Эти команды сбрасывают бит FC в 0.

Результат логической операции

Бит 1 слова состояния называется битом результата логической операции (бит RLO, см. рис. 2–6). Этот бит хранит результат цепи битовых логических операций или операций сравнения. Изменения сигнального состояния бита RLO может дать информацию, относящуюся к потоку сигнала.

Например, первая команда в сегменте контактного плана опрашивает состояние сигнала контакта и выдает результат 1 или 0. Эта команда сохраняет результат этого опроса состояния сигнала в бите RLO. Вторая команда в цепи битовых логических операций также опрашивает сигнальное состояние контакта и выдает результат опроса. Затем команда сопрягает этот результат со значением, хранящимся в бите RLO слова состояния, в соответствии с принципами булевой логики (см. выше Первичный опрос и главу 4). Результат этой логической операции сохраняется в бите RLO слова состояния, заменяя предыдущее значение, хранившееся в бите RLO. Каждая следующая команда в цепи выполняет логическую операцию с двумя значениями: результатом, полученным при опросе командой контакта, и текущим значением RLO.

Например, вы можете использовать булеву битовую логическую команду после первичного опроса для присваивания состояния содержимого ячейки памяти биту RLO или для запуска перехода.

Бит состояния

Бит 2 слова состояния называется битом состояния (бит STA, см. рис. 2–6). Бит состояния сохраняет значение того бита, к которому производится обращение. Состояние битовой команды, имеющей доступ к памяти для чтения (“Нормально открытый контакт”, “Нормально замкнутый контакт”), всегда равно значению бита, который эта операция опрашивает (бит, с которым она выполняет логическую операцию). Состояние битовой команды, имеющей доступ к памяти для записи (“Установить выход”, “Сбросить выход”, “Выходная катушка”) равно значению бита, в который команда производит запись, или, если запись не производится, значению бита, к которому команда обращается. Бит состояния не имеет значения для битовых команд, которые не обращаются к памяти. Эти команды устанавливают бит состояния в 1 (STA=1). Бит состояния не опрашивается командой. Он анализируется только при тестировании программы (статус программы).

Бит OR

Бит 3 слова состояния называется битом OR (см. рис. 2–6). Бит OR необходим, если вы используете команды над контактами для выполнения логической операции И перед ИЛИ. Логические операции ИЛИ соответствуют параллельному включению контактов. Логические операции И соответствуют последовательному включению контактов (см. раздел 2.2). Функция И может содержать следующие команды: “Нормально открытый контакт” и “Нормально замкнутый контакт”. Бит OR показывает этим операциям, что ранее выполненная функция И дала значение “1”, тем самым предвосхищая результат логической операции ИЛИ. Любая другая команда, обрабатывающая биты, сбрасывает бит OR.

Бит переполнения

Бит 5 слова состояния называется битом переполнения (бит OV, см. рис. 2–6). Бит OV указывает на наличие ошибки. Он устанавливается арифметической операцией или операцией сравнения для чисел с плавающей точкой, после возникновения ошибки (переполнение, недопустимая операция, недопустимое число с плавающей точкой). Бит устанавливается (в случае ошибки) или сбрасывается в соответствии с результатом арифметической операции или операции сравнения.

Бит сохраняемого переполнения

Бит 4 слова состояния называется битом сохраняемого переполнения (бит OS, см. рис. 2–6). Бит OS устанавливается вместе с битом OV при возникновении ошибки. Так как бит OS остается установленным после устранения ошибки (в отличие от бита OV), то он указывает, произошла ли ошибка в одной из ранее выполненных операций. Бит OS сбрасывают следующие команды: JOS (Перейти после сохраненного переполнения, программирование на STL), вызовы блоков и конец блока.

Код условия 1 и Код условия 0

Биты 7 и 6 слова состояния называются кодом условия 1 и кодом условия 0 (СС 1 и СС 0, см. рис. 2–6). СС 1 и СС 0 дают информацию о следующих результатах или битах:

- результат арифметической операции
- результат операции сравнения
- результат цифровой операции
- биты, которые были выдвинуты из операнда операцией сдвига или циклического сдвига

Таблицы от 2–10 до 2–15 перечисляют значения СС 1 и СС 0, после того как ваша программа выполняет определенные команды.

Таблица 2–10. СС 1 и СС 0 после арифметических операций, без переполнения		
СС 1	СС 0	Объяснение
0	0	Результат = 0
0	1	Результат < 0
1	0	Результат > 0

Таблица 2–11. СС 1 и СС 0 после арифметических операций с целыми числами, с переполнением		
СС 1	СС 0	Объяснение
0	0	Переполнение отрицательной области при сложении целых и двойных целых чисел
0	1	Переполнение отрицательной области при умножении целых и двойных целых чисел Переполнение положительной области при сложении и вычитании целых чисел, сложении и вычитании двойных целых чисел, при получении дополнения до двух целого числа и двойного целого числа
1	0	Переполнение положительной области при умножении целых и двойных целых чисел, при делении целых и двойных целых чисел Переполнение отрицательной области при сложении и вычитании целых чисел, сложении и вычитании двойных целых чисел
1	1	Деление на 0 при делении целых и двойных целых чисел и при получении остатка от деления двойных целых чисел

Таблица 2–12. СС 1 и СС 0 после арифметических операций над числами с плавающей точкой, с переполнением		
СС 1	СС 0	Объяснение
0	0	Поэтапная потеря значимости
0	1	Переполнение отрицательной области
1	0	Переполнение положительной области
1	1	Недопустимая операция

Таблица 2–13. CC 1 и CC 0 после операций сравнения

CC 1	CC 0	Объяснение
0	0	IN2 = IN1
0	1	IN2 < IN1
1	0	IN2 > IN1
1	1	IN1 или IN2 – недопустимое число с плавающей точкой

Таблица 2–14. CC 1 и CC 0 после операций сдвига и циклического сдвига

CC 1	CC 0	Объяснение
0	0	Последний выдвинутый бит = 0
1	0	Последний выдвинутый бит = 1

Таблица 2–15. CC 1 и CC 0 после логических операций со словами

CC 1	CC 0	Объяснение
0	0	Результат = 0
1	0	Результат <> 0

Бит двоичного результата

Бит 8 слова состояния называется битом двоичного результата (бит BR, см. рис. 26). Бит BR образует связь между обработкой битов и слов. Этот бит позволяет вашей программе интерпретировать результат операции над словами как двоичный результат и встроить этот результат в цепь двоичных логических операций. С этой точки зрения, BR представляет собой внутримашинный меркер, в котором сохраняется RLO перед изменяющей RLO операцией со словами, так что RLO остается доступным для продолжения прерванной цепи битов после того, как операция выполнена. Например, бит BR дает вам возможность программировать функциональный блок (FB) или функцию (FC) в списке команд (AWL) и вызывать этот FB или эту FC из контактного плана (KOP).

Когда вы пишете функциональный блок или функцию, которые хотите вызвать из KOP, безразлично, пишете ли Вы FB или FC в AWL или KOP, вы отвечаете за манипулирование с битом BR. Бит BR соответствует разрешающему выходу (ENO) блока KOP. Вы должны сохранить RLO в бите BR командой SAVE (в AWL) или с помощью катушки --- (SAVE) (в KOP) в соответствии со следующими критериями:

- сохраните RLO равным 1 в бите BR для случая, когда FB или FC обрабатывается без ошибок
- сохраните RLO равным 0 в бите BR для случая, когда при обработке FB или FC происходит ошибка.

Вы должны запрограммировать эти команды в конце FB или FC, так чтобы они выполнялись как последние команды в блоке.



Предупреждение

Возможен непреднамеренный сброс бита BR в 0.

Если вы пишете FB или FC в KOP и обрабатываете бит BR не так, как описано выше, то FB или FC может переписать бит BR другого FB или другой FC.

Во избежание этой ошибки сохраняйте RLO в конце FB или FC, как описано выше.

Значение EN/ENO

Параметры разрешающего входа (EN) и разрешающего выхода (ENO) блока KOP функционируют в соответствии со следующими принципами:

- Если EN не активизирован (т.е. имеет состояние сигнала, равное 0), то блок свою функцию не выполняет, и ENO не активизируется (т.е. его состояние сигнала тоже равно 0).
- Если EN активизирован (т.е. имеет состояние сигнала, равное 1) и соответствующий блок выполняет свою функцию без ошибок, то ENO тоже активизируется (т.е. его состояние сигнала тоже равно 1).
- Если EN активизирован (т.е. он имеет состояние сигнала, равное 1) и при обработке функции возникает ошибка, то ENO не активизируется (т.е. его состояние сигнала равно 0).

Если вы в своей программе вызываете системный функциональный блок (SFB) или системную функцию (SFC), то SFB или SFC указывает, выполнил ли CPU эту функцию без ошибок или с ошибкой, предоставляя следующую информацию в бите двоичного результата:

- Если при исполнении возникает ошибка, то бит BR равен 0.
- Если функция была выполнена без ошибок, то бит BR равен 1.

3 Адресация

Обзор главы

Раздел	Описание	Стр.
3.1	Обзор	3–2
3.2	Виды адресов	3–4

3.1 Обзор

Что такое адресация?

Многие команды КОР работают с одним или несколькими адресами (операндами). Этот операнд задает константу или место, где команда находит переменную, с которой она выполняет логическую операцию. Это место может быть битом, байтом, словом или двойным словом.

Возможными операндами, например, являются:

- константа, значение таймера или счетчика или цепочка символов ASCII
- бит в слове состояния программируемого логического контроллера
- блок данных и ячейка памяти внутри области блока данных

Непосредственная и прямая адресация

В вашем распоряжении имеются следующие виды адресации:

- Непосредственная адресация (задание константы в качестве операнда)
- Прямая адресация (задание переменной в качестве операнда)

Рис. 3–1 показывает пример непосредственной и прямой адресации.

Функция блока состоит в том, чтобы сравнить два входных параметра (в данном случае два целых числа), чтобы установить, меньше или равен первый вход второму. Константа 50 вводится как входной параметр IN1. Меркерное слово MW200, адрес в памяти, вводится как входной параметр IN2.

Так как в этом примере константа 50 является фактическим значением, с которым IN1 блока должен работать, то 50 является непосредственным адресом. Так как MW200 указывает на место в памяти, где находится значение, с которым должен работать IN2 блока, MW200 является прямым адресом. MW200 - это адрес, а не фактическое значение само по себе.

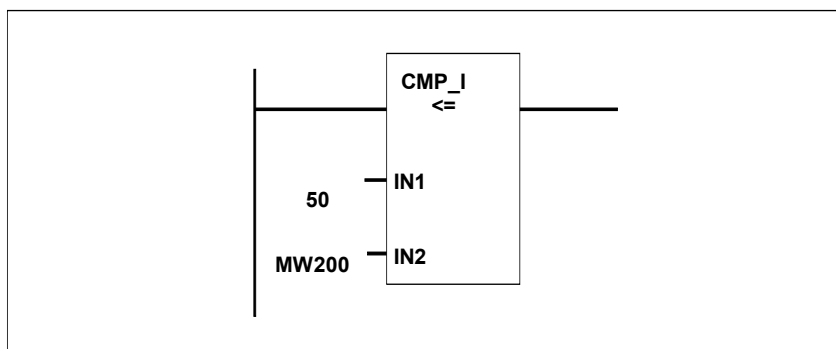


Рис. 3–1. Непосредственная и прямая адресация

Таблица 3–1. Форматы констант для непосредственной адресации, использующей элементарные типы данных

Тип и описание	Размер в битах	Возможные форматы	Диапазон и представление чисел (от минимального до максимального значения)	Пример
BOOL (бит)	1	Булев текст	TRUE/FALSE	TRUE
BYTE (байт)	8	Шестнадцатеричное число	В от В#16#0 до В#16#FF	В#16#10 byte#16#10
WORD (слово)	16	Двоичное число Шестнадцатеричное число BCD Десятичное число без знака	от 2#0 до 2#1111_1111_1111_1111 от W#16#0 до W#16#FFFF от С#0 до С#999 от В#(0,0) до В#(255,255)	2#0001_0000_0000_0000 W#16#1000 word16#1000 С#998 В#(10,20) byte#(10,20)
DWORD (двойное слово)	32	Двоичное число Шестнадцатеричное число Десятичное число без знака	от 2#0 до 2#1111_1111_1111_1111_1111_1111_1111_1111 от DW#16#0000_0000 до DW#16#FFFF_FFFF от В#(0,0,0,0) до В#(255,255,255,255)	2#1000_0001_0001_1000_1011_1011_0111_1111 DW#16#00A2_1234 dword#16#00A2_1234 В#(1,14,100,120) byte#(1,14,100,120)
INT (целое число)	16	Десятичное число со знаком	от –32768 до 32767	1
DINT (двойное целое число)	32	Десятичное число со знаком	от L#-2147483648 до L#2147483647	L#1
REAL (число с плавающей точкой)	32	Число с плавающей точкой в формате IEEE	Верхняя граница: ±3.402823е+38 Нижняя граница: ±1.175495е–38	1.234567е+13
S5TIME (время SIMATIC)	16	Время S5 шагами по 10 мс (как значение по умолчанию)	от S5T#0H_0M_0S_10MS до S5T#2H_46M_30S_0MS и S5T#0H_0M_0S_0MS	S5T#0H_1M_0S_0MS S5TIME#0H_1M_0S_0MS
TIME (время IEC)	32	Время IEC шагами по 1 мс, целое число со знаком	от T#–24D_20H_31M_23S_648MS до T#24D_20H_31M_23S_647MS	T#0D_1H_1M_0S_0MS TIME#0D_1H_1M_0S_0MS
DATE (дата IEC)	16	Дата IEC шагами по 1 дню	от D#1990–1–1 до D#2168–12–31	D#1994–3–15 DATE#1994–3–15
TIME_OF_DAY (время суток)	32	Время суток шагами по 1 мс	от TOD#0:0:0.0 до TOD#23:59:59.999	TOD#1:10:3.3 TIME_OF_DAY#1:10:3.3
CHAR (символ)	8	Character	'A','B' и т.д.	'E'

3.2 Виды адресов

Возможные адреса

Адрес команды КОР может указывать на один из следующих объектов:

- бит, состояние сигнала которого должно быть опрошено
- бит, которому присваивается состояние сигнала цепи логических операций
- бит, которому присваивается результат логической операции (RLO)
- бит, который должен быть установлен или сброшен
- число, указывающее счетчик, который должен быть увеличен или уменьшен
- число, указывающее, какой таймер должен быть использован
- бит памяти (меркер) фронта, сохраняющий предыдущий результат логической операции (RLO)
- бит памяти (меркер) фронта, сохраняющий предыдущее состояние сигнала другого операнда
- байт, слово или двойное слово, содержащее значение, с которым должен работать элемент и блок КОР байт, слово или двойное слово, содержащее значение, с которым должен работать элемент и блок КОР.
- номер блока данных (DB или DI), который должен быть открыт или создан
- номер подлежащей вызову функции (FC), системной функции (SFC), функционального блока (FB) или системного функционального блока (SFB)
- метка, на которую нужно перейти

Идентификаторы адресов

Переменные, используемые в качестве адресов, состоят из идентификатора адреса и адреса внутри области памяти, заданной в идентификаторе адреса. Идентификатор адреса может принадлежать к одному из следующих двух видов:

- Идентификатор адреса, указывающий следующее:
 - Область памяти, в которой команда находит значение (объект данных), с которым необходимо выполнить операцию (например, I для области памяти входов образа процесса, см. таблицу 2–5)
 - Размер объекта данных, с которым команда должна выполнить операцию (например, B для байта, W для слова и D для двойного слова, см. таблицу 2–5)
- Идентификатор адреса, указывающий область памяти, но не размер объекта данных в этой области (например, идентификатор, указывающий область T для таймера, S для счетчика, DB или DI для блока данных, плюс номер таймера, счетчика или блока данных, см. таблицу 2–5).

Указатели

Указатель – это элемент, распознающий местоположение переменной. *variable*. Указатель содержит адрес вместо значения. При назначении фактического параметра параметрическому типу «pointer [указатель]» вы указываете адрес в памяти. STEP 7 дает возможность вводить указатель или в формате указателя, или просто как адрес (например, M 50.0). Далее следует пример формата указателя для обращения к данным, начинающимся с M 50.0:

P#M50.0

Работа со словом или двойным словом как с объектом данных

Если вы работаете с командой, идентификатор адреса которой задает область памяти вашего программируемого логического контроллера, и с объектом данных, который по своему размеру является словом или двойным словом, то вы должны принять во внимание, что на адрес памяти всегда ссылаются как на байтовый адрес. Этот байтовый адрес является самым малым номером байта или номером старшего байта. Например, адрес в операторе, показанном на рис. 3–2, ссылается на четыре последовательных байта в области памяти M, начиная с байта 10 (MB10) и вплоть до байта 13 (MB13).

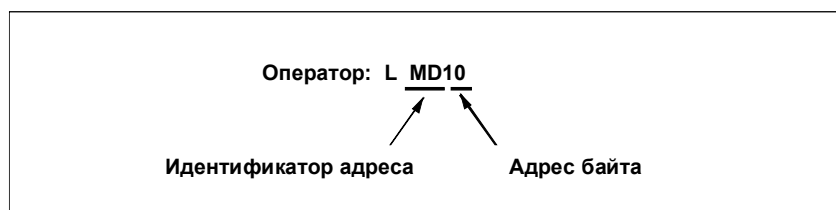


Рис. 3–2. Пример ссылки на адрес в памяти как адрес байта

На рис. 3–3 показаны объекты данных следующих размеров:

- Двойное слово: двойное слово памяти MD10
- Слово: слова памяти MW10, MW11 и MW12
- Байт: байты памяти MB10, MB11, MB12 и MB13

Если вы используете абсолютные адреса размером в слово или двойное слово, то убедитесь, что вы избежали таких назначений байтов, при которых они перекрываются.

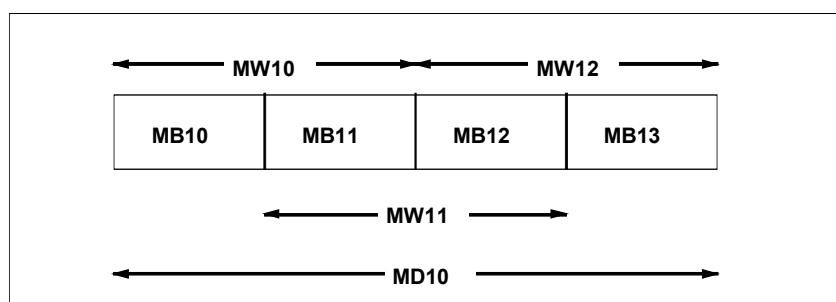


Рис. 3–3. Ссылка на адрес памяти как на байтовый адрес

4 Битовые логические операции

Обзор главы

Раздел	Описание	Стр.
4.1	Обзор	4–2
4.2	Нормально открытый контакт	4–3
4.3	Нормально замкнутый контакт	4–4
4.4	Выходная катушка	4–5
4.5	Промежуточный выход (коннектор)	4–6
4.6	Инвертирование результата логической операции	4–7
4.7	Сохранение RLO в регистре BR	4–8
4.8	Установка выхода	4–9
4.9	Сброс выхода	4–10
4.10	Установка начального значения счетчика	4–11
4.11	Катушка со счетчиком прямого счета	4–12
4.12	Катушка со счетчиком обратного счета	4–13
4.13	Катушка с таймером – формирователем импульса	4–14
4.14	Катушка с таймером – формирователем удлиненного импульса	4–15
4.15	Катушка с таймером – формирователем задержки включения	4–16
4.16	Катушка с таймером – формирователем задержки включения с запоминанием	4–17
4.17	Катушка с таймером – формирователем задержки выключения	4–18
4.18	Обнаружение положительного фронта RLO	4–19
4.19	Обнаружение отрицательного фронта RLO	4–20
4.20	Обнаружение положительного фронта сигнала	4–21
4.21	Обнаружение отрицательного фронта сигнала	4–22
4.22	Установка-сброс триггера	4–23
4.23	Сброс-установка триггера	4–24

4.1 Обзор

Объяснение

Битовые логические операции работают с двумя цифрами: 1 и 0. Эти две цифры образуют основание системы счисления, называемой двоичной системой. Две цифры 1 и 0 называются двоичными цифрами или битами. В мире контактов и катушек 1 означает активное состояние или протекание тока, а 0 – неактивное состояние или отсутствие протекания тока.

Битовые логические операции интерпретируют сигнальные состояния 1 и 0 и сопрягают их в соответствии с правилами булевой логики. Эти сопряжения дают результат 1 или 0, который называется «результатом логической операции» (RLO, см. раздел 2.3). Логические операции, запускаемые битовыми логическими командами, выполняют ряд функций.

Функции

Имеются битовые логические команды для выполнения следующих функций:

- Нормально открытый и нормально замкнутый контакт опрашивают состояние сигнала контакта и дают результат, который или копируется в бит результата логической операции (RLO), или логически сопрягаются с RLO. Если эти контакты соединены последовательно, то они сопрягают результат опроса состояния сигнала в соответствии с таблицей истинности для логической функции И (см. таблицу 2–8); если они соединены параллельно, то они сопрягают свой результат в соответствии с таблицей истинности для логической функции ИЛИ (см. таблицу 2–9).
- Выходная катушка и промежуточный выход (коннектор) присваивают RLO или временно запоминают его.
- Следующие команды реагируют на RLO, равный 1:
 - установка выхода и сброс выхода
 - триггеры "Установка-сброс" и "Сброс-установка"
- Другие команды реагируют на положительный или отрицательный фронт для выполнения следующих функций:
 - увеличение и уменьшение значения счетчика
 - запуск таймера
 - создание выхода, равного 1
- Остальные команды воздействуют на RLO непосредственно следующими способами:
 - отрицание (инвертирование) RLO
 - сохранение RLO в бите двоичного результата слова состояния

В данной главе катушки со счетчиками и таймерами представляются в формате SIMATIC и международном (английском) формате.

4.2 Нормально открытый контакт

Описание

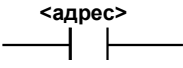
Вы можете использовать команду *Нормально открытый контакт (адрес)* для опроса состояния сигнала контакта по указанному адресу. Если состояние сигнала по указанному адресу равно 1, то контакт замкнут, и команда дает результат, равный 1. Если состояние сигнала по указанному адресу равно 0, то контакт разомкнут, и команда дает результат, равный 0.

Когда *Нормально открытый контакт (адрес)* является первой командой в логической цепи, эта команда сохраняет результат опроса сигнала в бите результата логической операции (RLO).

Любая команда *Нормально открытый контакт (адрес)*, не являющаяся первой в логической цепи, сопрягает результат опроса состояния сигнала со значением, хранящимся в бите RLO. Эта команда формирует сопряжение одним из следующих двух способов:

- Если команда используется в последовательном соединении, то она сопрягает результат опроса состояния сигнала в соответствии с таблицей истинности логической функции И.
- Если команда используется в параллельном соединении, то она сопрягает результат опроса состояния сигнала в соответствии с таблицей истинности логической функции ИЛИ.

Таблица 4–1. Элемент "Нормально открытый контакт (адрес)" и параметр

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	<адрес>	BOOL TIMER COUNTER	I, Q, M, T, C, D, L	Адрес указывает бит, сигнальное состояние которого опрашивается.

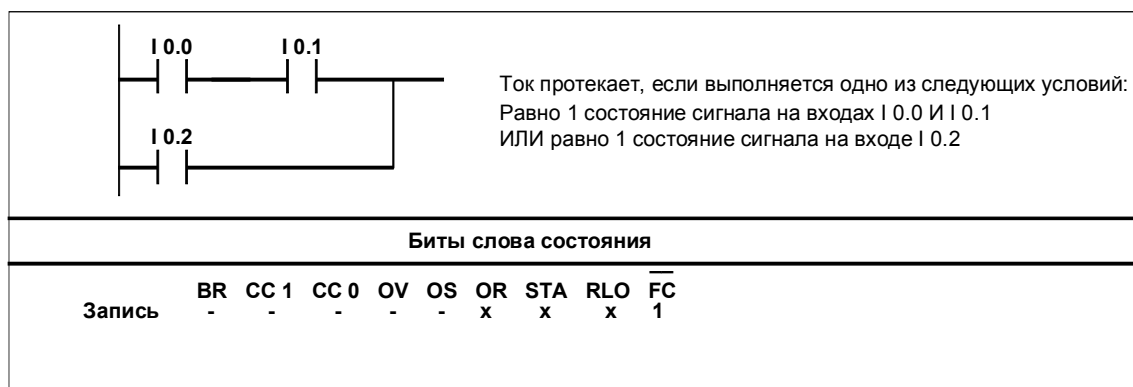


Рис. 4–1. Нормально открытый контакт (адрес)


4.3 Нормально замкнутый контакт

Описание

Вы можете использовать команду *Нормально замкнутый контакт (адрес)* для опроса состояния сигнала контакта по указанному адресу. Если состояние сигнала по указанному адресу равно 0, то контакт замкнут, и команда дает результат, равный 1. Если состояние сигнала по указанному адресу равно 1, то контакт разомкнут, и команда дает результат, равный 0. Когда *Нормально замкнутый контакт (адрес)* является первой командой в логической цепи, эта команда сохраняет результат опроса сигнала в бите результата логической операции (RLO).

Любая команда *Нормально замкнутый контакт (адрес)*, не являющаяся первой в логической цепи, сопрягает результат опроса состояния сигнала со значением, хранящимся в бите RLO. Эта команда формирует сопряжение одним из следующих двух способов:

- Если команда используется в последовательном соединении, то она сопрягает результат опроса состояния сигнала в соответствии с таблицей истинности логической функции И.
- Если команда используется в параллельном соединении, то она сопрягает результат опроса состояния сигнала в соответствии с таблицей истинности логической функции ИЛИ.

Таблица 4–2. Элемент "Нормально замкнутый контакт (адрес)" и параметр				
Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	< адрес >	BOOL TIMER COUNTER	I, Q, M, T, C, D, L	Адрес указывает бит, сигнальное состояние которого опрашивается.

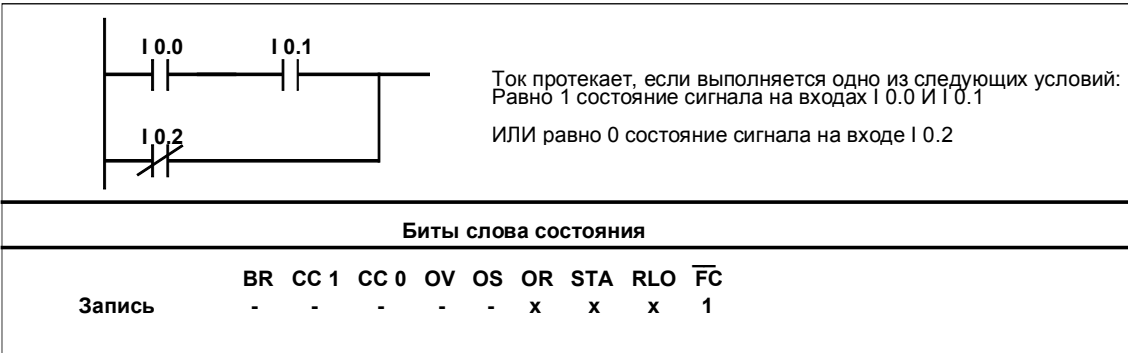


Рис. 4–2. Нормально замкнутый контакт (адрес)

4.4 Выходная катушка

Описание

Команда *Выходная катушка* работает подобно катушке в релейно-контактной схеме. Катушка в конце цепи пропускает или не пропускает ток в зависимости от следующих критериев:

- Если ток может протекать через цепь вплоть до катушки (т.е. состояние сигнала цепи равно 1), то катушка пропускает ток.
- Если ток не может протекать по всей цепи вплоть до катушки (т.е. состояние сигнала цепи равно 0), то катушка не пропускает ток.

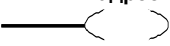
Цепь логических операций представляет цепь тока. Команда *Выходная катушка* присваивает состояние сигнала логической цепи КОР катушке, к которой адресуется команда (это то же самое, что и присвоение состояния сигнала бита RLO операнду). Если ток течет через цепь, то состояние сигнала логической цепи равно 1; в противном случае состояние сигнала равно 0.

Команда *Выходная катушка* испытывает воздействие со стороны *Главного управляющего реле* (Master Control Relay, MCR). Дополнительную информацию о том, как функционирует MCR, можно получить в разделе 16.5.

Выходную катушку можно поместить только на правом конце логической цепи. Возможно использование нескольких Выходных катушек. *Выходную катушку* нельзя помещать одну в пустой сегмент. Катушка должна иметь предшествующее соединение.

Вы можете создать инвертированный выход с помощью команды *Инвертировать результат логической операции*.

Таблица 4–3. Элемент "Выходная катушка" и параметр

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	< адрес >	BOOL	I, Q, M, D, L	Адрес указывает бит, которому присваивается сигнальное состояние логической цепи.

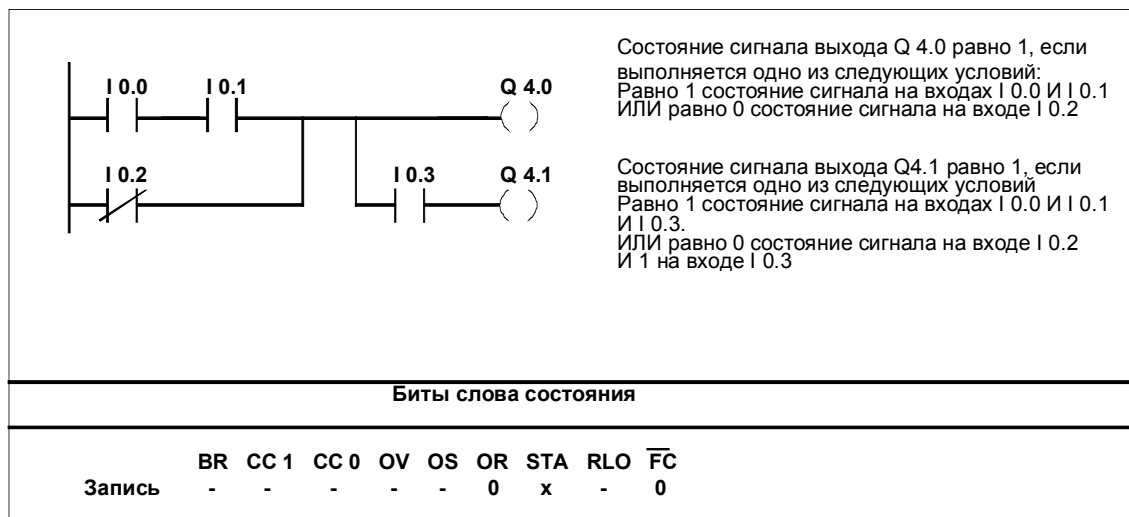


Рис. 4–3. Выходная катушка

4.5 Промежуточный выход (коннектор)

Описание

Команда *Промежуточный выход (коннектор)* – это промежуточный присваивающий элемент, сохраняющий RLO. Этот промежуточный присваивающий элемент запоминает битовую логическую комбинацию последней открытой ветви перед этим элементом. При последовательном соединении с другими контактами *Промежуточный выход (коннектор)* действует как обычный контакт.

Команда *Промежуточный выход (коннектор)* испытывает воздействие со стороны *Главного управляющего реле* (Master Control Relay, MCR). Дополнительную информацию о том, как функционирует MCR, можно получить в разделе 16.5.

На размещение *Промежуточного выхода* накладываются определенные ограничения. Например, элемент *Промежуточный выход (коннектор)* никогда не может находиться в конце сегмента или в конце открытой ветви. См. также раздел 2.1.

Вы можете создать инвертированный выход с помощью команды *Инвертировать результат логической операции*.

Таблица 4–4. Элемент "Промежуточный выход (коннектор)" и параметр				
Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	< адрес >	BOOL	I, Q, M, D, L ¹	Адрес указывает бит, которому присваивается RLO.

¹ Для команды *Промежуточный выход (коннектор)* можно использовать только адрес из области памяти L, если вы описываете его в VAR_TEMP. С этой командой вы не можете использовать область памяти L для абсолютной адресации.

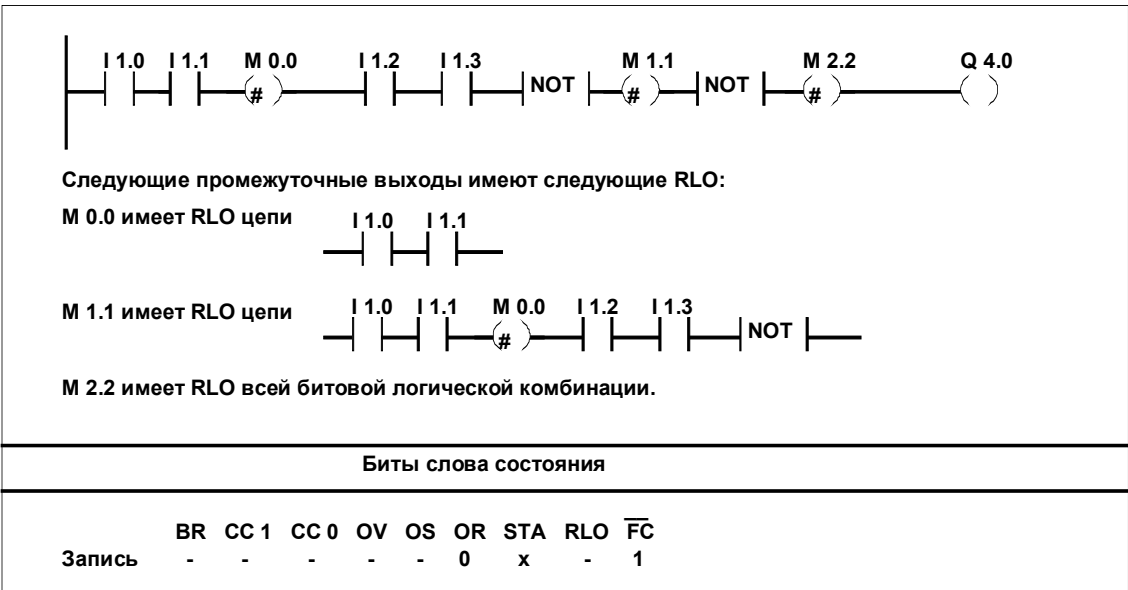



Рис. 4–4. Промежуточный выход (коннектор)

4.6 Инвертирование результата логической операции

Описание

Команда *Инвертировать результат логической операции* выполняет отрицание RLO.

Таблица 4–5. Элемент "Инвертирование результата логической операции"				
Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	Нет	-	-	-

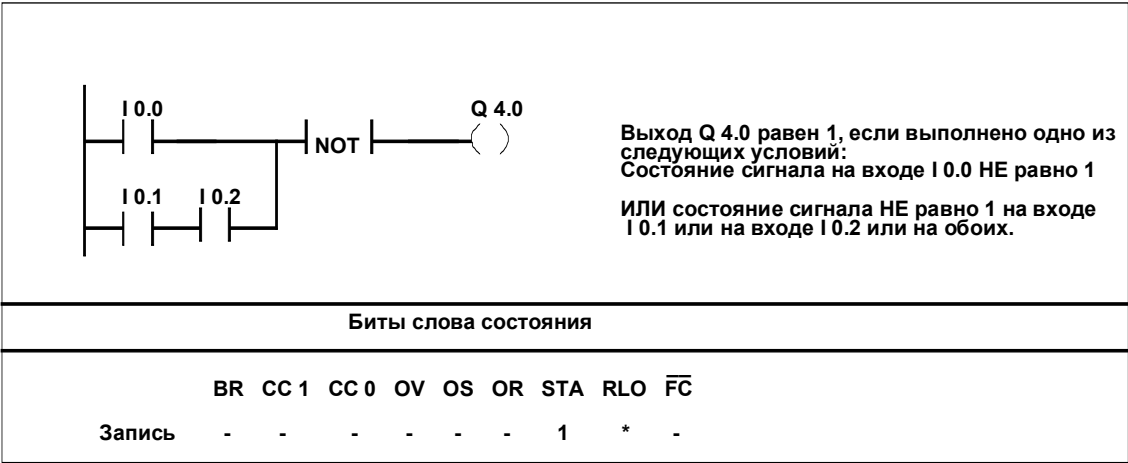


Рис. 4–5. Инвертирование результата логической операции

4.7 Сохранение RLO в регистре BR

Описание

Команда *Сохранить RLO в регистре BR* сохраняет RLO в бите BR слова состояния. Бит первичного опроса /FC не сбрасывается.

По этой причине состояние бита BR включается в логическую операцию И в следующем сегменте.

Мы не рекомендуем вам использовать SAVE, а затем опрашивать бит BR в том же блоке или в починенных блоках, так как бит BR может быть изменен многими командами в промежутке между этими событиями. Целесообразно использовать команду SAVE перед выходом из блока, так как выход ENO (=бит BR) тогда устанавливается на значение бита RLO, и вы можете после этого контролировать ошибки в блоке.

Таблица 4–6. Сохранение RLO в регистре BR				
Элемент KOP	Параметр	Тип данных	Область памяти	Описание
—(SAVE)	Нет	-	-	-

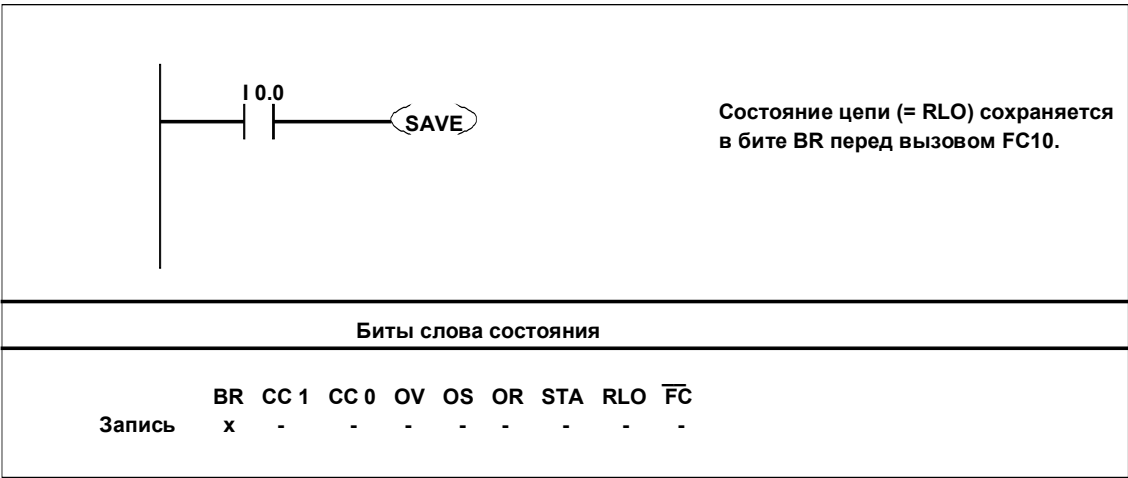


Рис. 4–6. Сохранение RLO в регистре BR

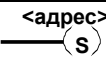
4.8 Установка выхода

Описание

Команда *Установка выхода* выполняется только тогда, когда $RLO = 1$. Если $RLO = 1$, эта команда устанавливает указанный адрес в 1. Если $RLO = 0$, команда не оказывает влияния на указанный адрес. Адрес остается неизменным.

Команда *Установка выхода* испытывает воздействие со стороны *Главного управляющего реле* (Master Control Relay, MCR). Дополнительную информацию о том, как функционирует MCR, можно получить в разделе 16.5.

Таблица 4–7. Элемент "Установка выхода" и параметр

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	< адрес >	BOOL	I, Q, M, D, L	Адрес указывает бит, который должен быть установлен.

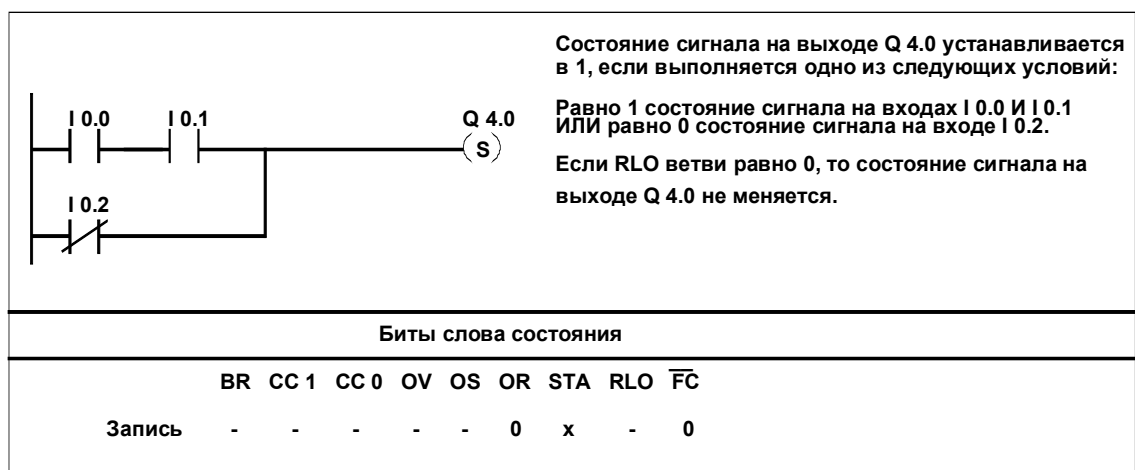


Рис. 4–7. Установка выхода

4.9 Сброс выхода

Описание

Команда *Сброс выхода* выполняется только тогда, когда RLO = 1. Если RLO = 1, эта команда сбрасывает указанный адрес в 0. Если RLO = 0, команда не оказывает влияния на указанный адрес. Адрес остается неизменным.

Команда *Сброс выхода* испытывает воздействие со стороны *Главного управляющего реле* (Master Control Relay, MCR). Дополнительную информацию о том, как функционирует MCR, можно получить в разделе 16.5.

Таблица 4–8. Элемент "Сброс выхода" и параметр				
Элемент КОР	Параметр	Тип данных	Область памяти	Описание
<div><адрес> — (R)</div>	< адрес >	BOOL TIMER COUNTER	I, Q, M, T, C, D, L	Адрес указывает бит, который должен быть сброшен.

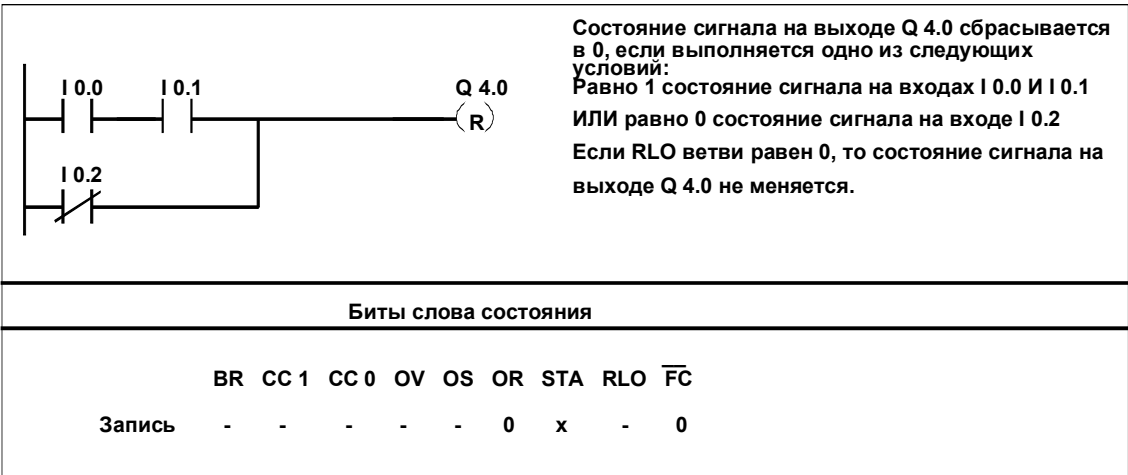


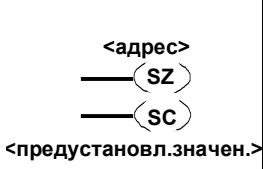
Рис. 4–8. Сброс выхода

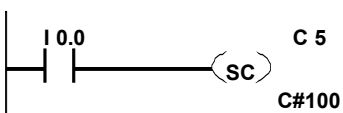
4.10 Установка начального значения счетчика

Описание

Вы можете использовать команду *Установить начальное значение счетчика* (SC), чтобы поместить предварительно заданное значение в указанный вами счетчик. Команда выполняется только при положительном фронте RLO (т.е. в RLO имеет место переход из 0 в 1).

Таблица 4–9. Элемент "Установка начального значения счетчика" и параметры, с сокращенным международным именем и сокращенным именем SIMATIC

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	Номер счетчика	COUNTER	C	Адрес указывает номер счетчика, в котором должно быть установлено начальное значение.
	Предустановленное значение	-	I, Q, M, D, L	Начальное значение может быть в диапазоне от 0 до 999. Значению должно предшествовать C#, указывающее на двоично-десятичный формат (BCD), например, C#100.



Если состояние сигнала на входе I 0.0 меняется с 0 на 1 (т.е. имеет место положительный фронт в RLO), в счетчике C 5 устанавливается начальное значение 100. C# указывает, что вы вводите значение в формате BCD.

Когда вы сохраните цепочку, это значение будет Представлено на вашем экране как w#16#100.

Если положительный фронт отсутствует, то значение счетчика C 5 не меняется.

Биты слова состояния										
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	\overline{FC}	
Запись	-	-	-	-	-	0	x	-	0	

Рис. 4–9. Установка начального значения счетчика

4.11 Катушка со счетчиком прямого счета

Описание

Команда *Катушка со счетчиком прямого счета (CU)* увеличивает значение указанного счетчика на единицу, если RLO имеет положительный фронт (т.е. в RLO имеет место переход с 0 на 1) и значение счетчика меньше 999. Если положительный фронт RLO отсутствует или счетчик уже достиг 999, то значение счетчика не изменяется.

Команда *Установить начальное значение счетчика* устанавливает значение счетчика (см. раздел 4.10).

Таблица 4–10. Элемент "Катушка со счетчиком прямого счета" и параметр, с сокращенным международным именем и сокращенным именем SIMATIC				
Элемент KOP	Параметр	Тип данных	Область памяти	Описание
<div><div><адрес></div><div><div>—</div><div>(ZV)</div></div><div><div>—</div><div>(CU)</div></div></div>	Номер счетчика	COUNTER	C	Адрес указывает номер счетчика, содержимое которого нужно увеличить.

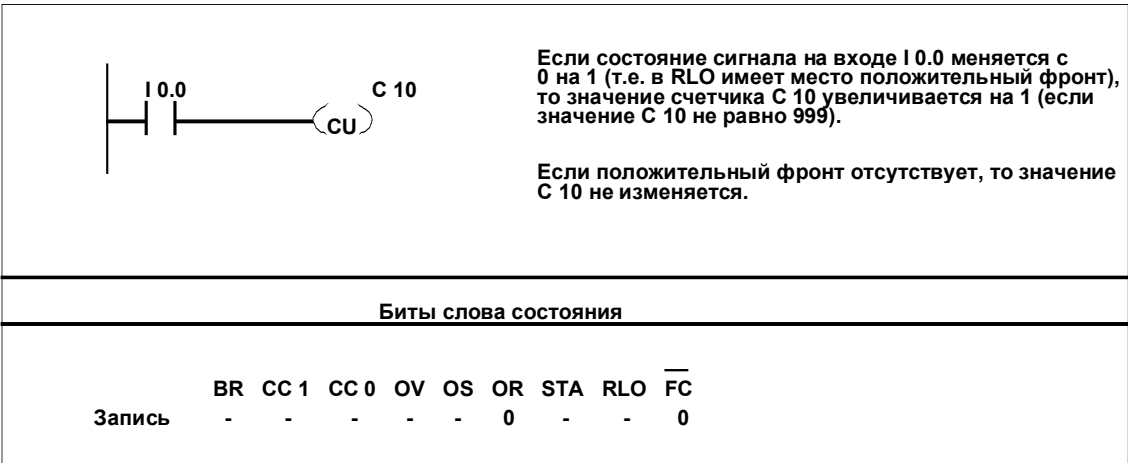


Рис. 4–10. Катушка со счетчиком прямого счета


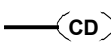
4.12 Катушка со счетчиком обратного счета

Описание

Команда *Катушка со счетчиком обратного счета (CD)* уменьшает значение указанного счетчика на единицу, если RLO имеет положительный фронт (т.е. в RLO имеет место переход с 0 на 1) и значение счетчика больше 0. Если положительный фронт RLO отсутствует или счетчик уже в 0, то значение счетчика не изменяется.

Команда *Установить начальное значение счетчика* устанавливает значение счетчика (см. раздел 4.10).

Таблица 4–11. Элемент "Катушка со счетчиком обратного счета" и параметр, с сокращенным международным именем и сокращенным именем SIMATIC

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
<p><адрес>  (ZR)</p> <p>  (CD)</p>	Номер счетчика	COUNTER	C	Адрес указывает номер счетчика, содержимое которого нужно уменьшить.

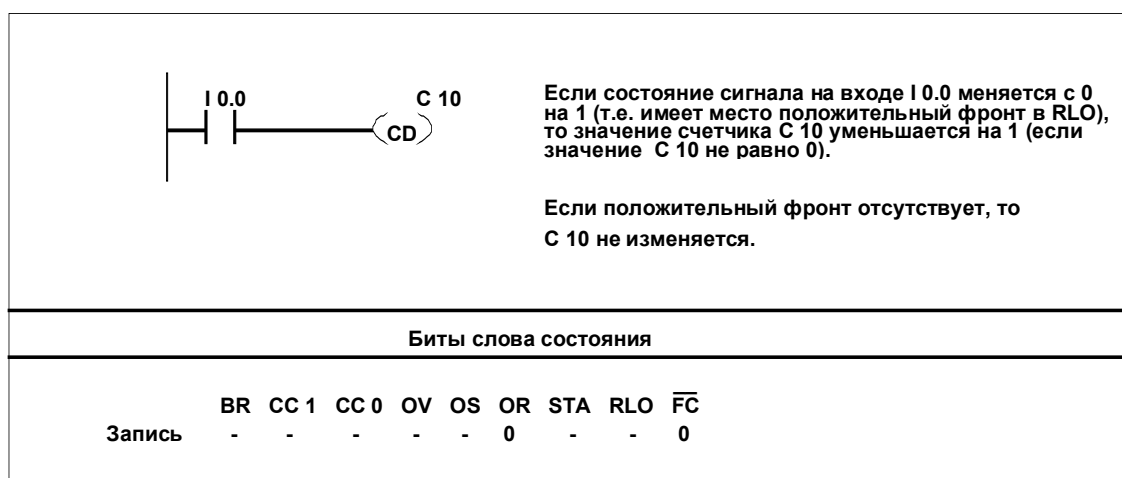


Рис. 4–11. Катушка со счетчиком обратного счета

4.13 Катушка с таймером – формирователем импульса

Описание

Команда *Катушка с таймером – формирователем импульса (SP)* запускает указанный таймер с заданным значением времени, если RLO имеет положительный фронт (т.е. в RLO имеет место переход с 0 на 1). Таймер продолжает работать с заданным временем, пока значение RLO положительно. Опрос состояния сигнала таймера на 1 дает результат, равный 1, пока таймер работает. Если RLO меняется с 1 на 0 до истечения заданного времени, то таймер останавливается. В этом случае опрос состояния сигнала на 1 дает результат, равный 0.

Единицами времени являются d (дни), h (часы), m (минуты), s (секунды) и ms (миллисекунды). За информацией о размещении таймера в памяти и его компонентах обратитесь к разделу 5.1.

Таблица 4–12. Элемент "Катушка с таймером – формирователем импульса" и параметры, с сокращенным международным именем и сокращенным именем SIMATIC				
Элемент KOP	Параметр	Тип данных	Область памяти	Описание
<div><адрес> —(SI) —(SP) <Значение времени></div>	Номер таймера	TIMER	T	Адрес указывает номер таймера, который должен быть запущен.
	Значение времени	S5TIME	I, Q, M, D, L	Значение времени (в формате S5TIME)

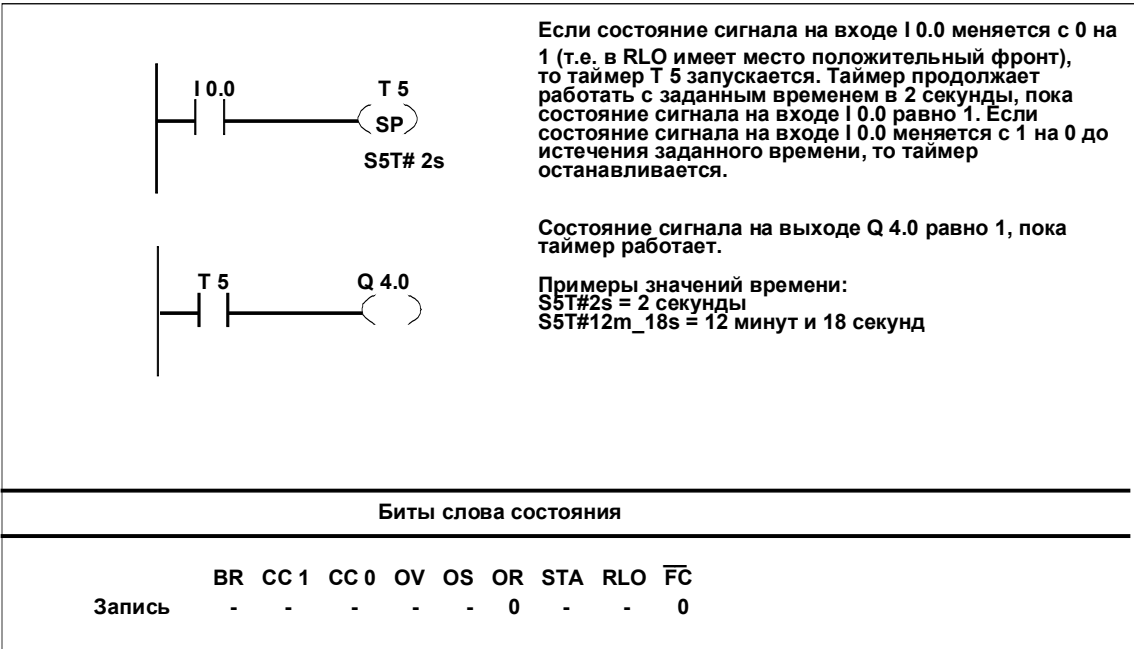


Рис. 4–12. Катушка с таймером – формирователем импульса

4.14 Катушка с таймером – формирователем удлиненного импульса

Описание

Команда *Катушка с таймером – формирователем удлиненного импульса* (SE) запускает указанный таймер с заданным значением времени, если RLO имеет положительный фронт (т.е. в RLO имеет место переход с 0 на 1). Таймер продолжает работать с заданным временем, даже если RLO становится равным 0 до истечения этого времени. Опрос состояния сигнала таймера на 1 дает результат, равный 1, пока таймер работает. Если RLO изменяется с 0 на 1 во время работы таймера, то таймер перезапускается с заданным временем. За информацией о размещении таймера в памяти и его компонентах обратитесь к разделу 5.1.

Таблица 4–13. Элемент "Катушка с таймером – формирователем удлиненного импульса" и параметры, с сокращенным международным именем и сокращенным именем SIMATIC

Элемент KOP	Параметр	Тип данных	Область памяти	Описание
	Номер таймера	TIMER	T	Адрес указывает номер таймера, который должен быть запущен.
	Значение времени	S5TIME	I, Q, M, D, L	Значение времени (в формате S5TIME)

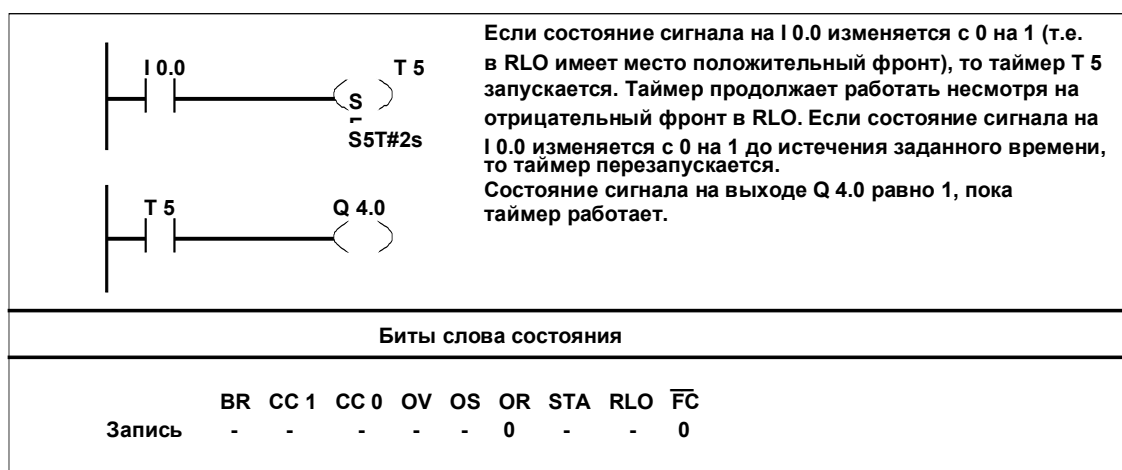


Рис. 4–13. Катушка с таймером – формирователем удлиненного импульса

4.15 Катушка с таймером – формирова­те­лем задержки включения

Описание

Команда *Катушка с таймером – формирова­те­лем задержки включения (SD)* запускает указанный таймер, если RLO имеет положительный фронт (т.е. в RLO имеет место переход с 0 на 1). Опрос состояния таймера на 1 дает результат, равный 1, когда указанное время истекло без ошибок, а RLO еще равен 1. Когда RLO изменяет свое значение с 1 на 0 во время работы таймера, таймер останавливается. В этом случае is case, а signal state с опрос состояния сигнала на 1 всегда дает результат, равный 0. За информацией о размещении таймера в памяти и его компонентах обратитесь к разделу 5.1.

Таблица 4–14. Элемент "Катушка с таймером – формирова­те­лем задержки включения" и параметры, с сокращенным международным именем и сокращенным именем SIMATIC

Элемент KOP	Параметр	Тип данных	Область памяти	Описание
<div><адрес> —(SE) —(SD) <Значение времени></div>	Номер таймера	TIMER	T	Адрес указывает номер таймера, который должен быть запущен.
	Значение времени	S5TIME	I, Q, M, D, L	Значение времени (в формате S5TIME)



Рис. 4–14. Катушка с таймером – формирова­те­лем задержки включения

4.17 Катушка с таймером – формирователем задержки выключения

Описание

Команда *Катушка с таймером – формирователем задержки выключения* (SF) запускает указанный таймер, если RLO имеет отрицательный фронт (т.е. в RLO имеет место переход с 1 на 0). Результат опроса состояния сигнала таймера на 1 равен 1, когда RLO равен 1 или когда таймер работает. Когда RLO изменяется с 0 на 1 во время работы таймера, таймер сбрасывается. Таймер не перезапускается, пока RLO не поменяет свое значение с 1 на 0. За информацией о размещении таймера в памяти и его компонентах обратитесь к разделу 5.1.

Таблица 4–16. Элемент "Катушка с таймером – формирователем задержки выключения" и параметры, с сокращенным международным именем и сокращенным именем SIMATIC				
Элемент KOP	Параметр	Тип данных	Область памяти	Описание
<div><адрес> —(SA) —(SF) <Значение времени></div>	Номер таймера	TIMER	T	Адрес указывает номер таймера, который должен быть запущен.
	Значение времени	S5TIME	I, Q, M, D, L	Значение времени (в формате S5TIME)

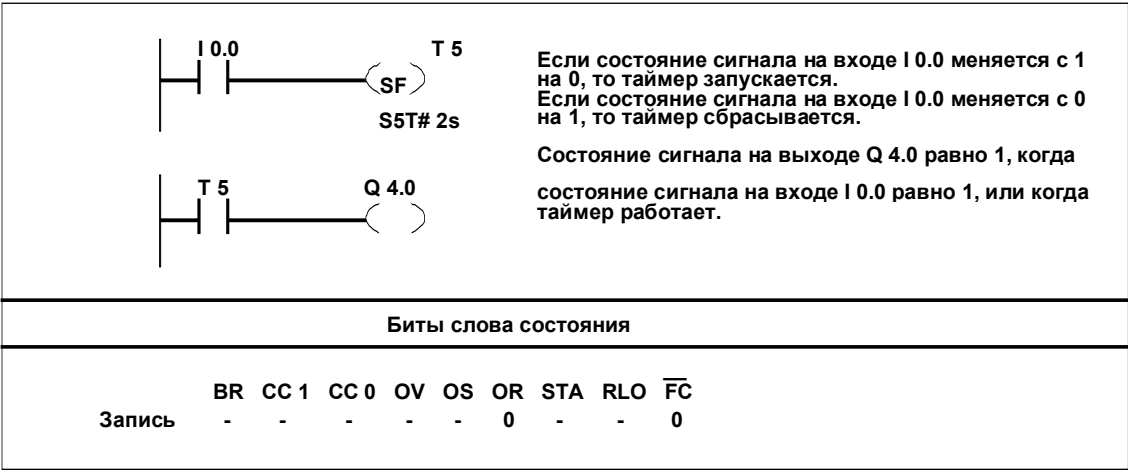


Рис. 4–16. Катушка с таймером – формирователем задержки выключения


4.18 Обнаружение положительного фронта RLO

Описание

Операция *Обнаружение положительного фронта RLO* распознает изменение во введенном адресе с 0 на 1 (нарастающий фронт) и отображает это после выполнения операции как $RLO = 1$. Текущее состояние сигнала в RLO сравнивается с состоянием сигнала адреса, бита памяти фронта. Если состояние сигнала адреса равно 0, а RLO перед операцией был равен 1, то RLO после операции будет равен 1 (импульс), и 0 во всех остальных случаях. Значение RLO перед операцией хранится в адресе.

На размещение элемента *Обнаружение положительного фронта RLO* накладываются определенные ограничения (см. раздел 2.1).

Таблица 4–17. Элемент "Обнаружение положительного фронта RLO" и параметр

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
<адрес 1> 	<адрес1>	BOOL	Q, M, D	Адрес указывает на бит памяти фронта, который хранит предыдущее значение RLO.

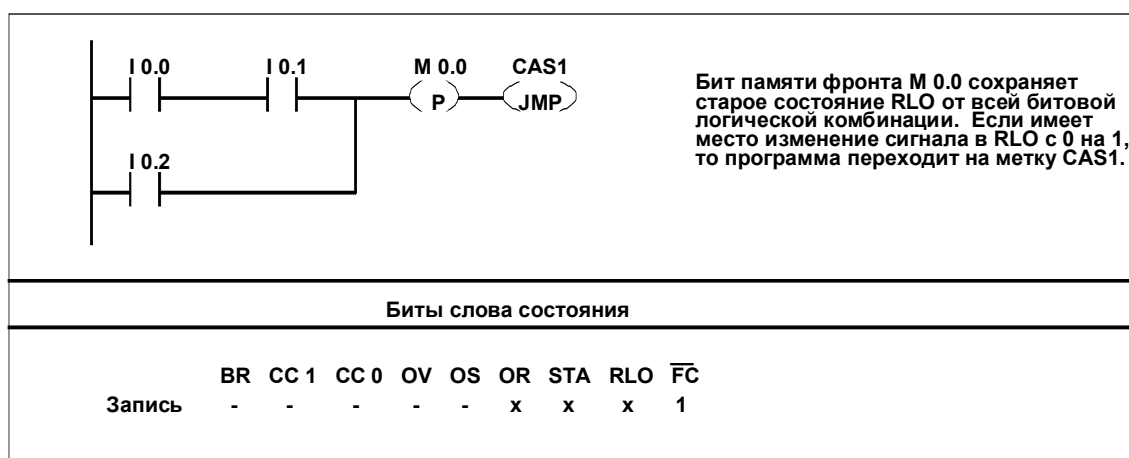


Рис. 4–17. Обнаружение положительного фронта RLO

4.19 Обнаружение отрицательного фронта RLO

Описание

Операция *Обнаружение отрицательного фронта RLO* распознает изменение во введенном адресе с 1 на 0 (падающий фронт) и отображает это после выполнения операции как RLO = 1. Текущее состояние сигнала в RLO сравнивается с состоянием сигнала адреса, бита памяти фронта. Если состояние сигнала адреса равно 1, а RLO перед операцией был равен 0, то RLO после операции будет равен 0 (импульс), и 1 во всех остальных случаях. Значение RLO перед операцией хранится в адресе.

На размещение элемента *Обнаружение отрицательного фронта RLO* накладываются определенные ограничения (см. раздел 2.1).

Таблица 4–18. Элемент "Обнаружение отрицательного фронта RLO" и параметр				
Элемент КОР	Параметр	Тип данных	Область памяти	Описание
<div><адрес1> —(N)—</div>	<адрес1>	BOOL	Q, M, D	Адрес указывает на бит памяти фронта, который хранит предыдущее значение RLO.

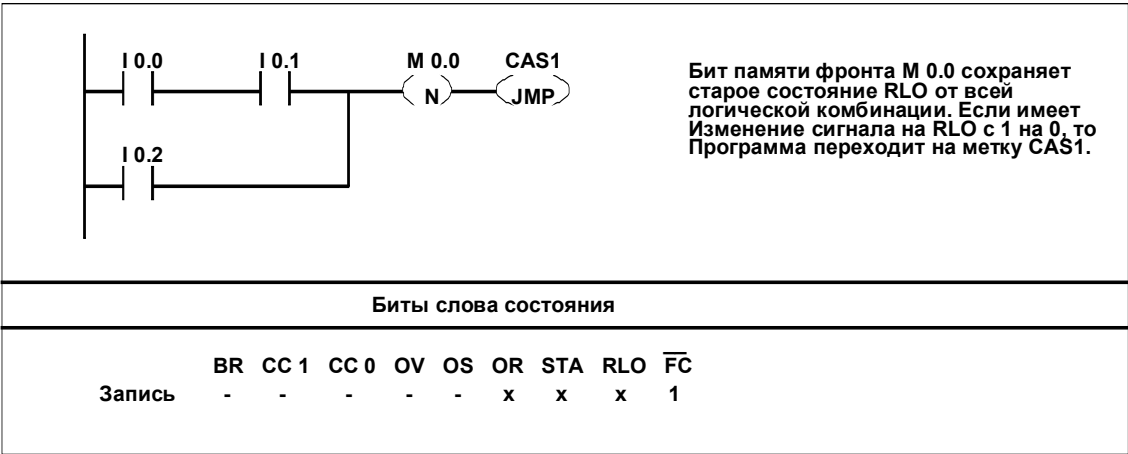


Рис. 4–18. Обнаружение отрицательного фронта RLO

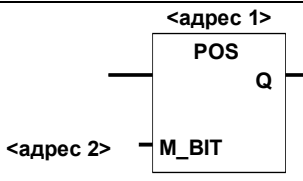
4.20 Обнаружение положительного фронта сигнала

Описание

Команда *Обнаружение положительного фронта сигнала* сравнивает состояние сигнала <адрес 1> с результатом опроса состояния сигнала, хранящимся в <адрес 2>. Если имеет место переход с 0 на 1, то выход Q равен 1. В противном случае он равен 0.

На размещение блока *Обнаружение положительного фронта сигнала* накладываются определенные ограничения (см. раздел 2.1).

Таблица 4–19. Блок "Обнаружение положительного фронта сигнала" и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	<адрес1>	BOOL	I, Q, M, D, L	Сигнал, подлежащий контролю на появление положительного фронта.
	M_BIT	BOOL	Q, M, D	Адрес M_BIT указывает бит памяти фронта, который хранит предыдущее состояние сигнала POS. Используйте для M_BIT область памяти входов образа процесса (I) только в том случае, если этот адрес уже не занят ни одним из модулей ввода.
	Q	BOOL	I, Q, M, D, L	Выход с однократным импульсом.

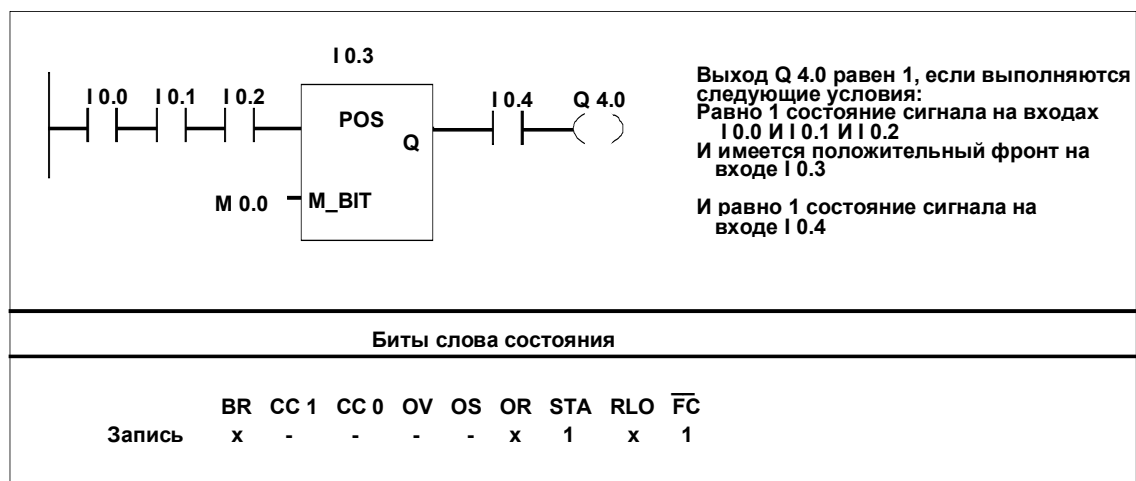


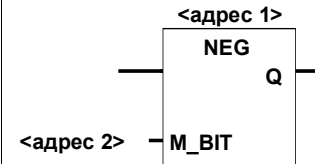
Рис. 4–19. Обнаружение положительного фронта сигнала

4.21 Обнаружение отрицательного фронта сигнала

Описание

Команда *Обнаружение отрицательного фронта сигнала* сравнивает состояние сигнала <адрес 1> с результатом опроса состояния сигнала, хранящимся в <адрес 2>. Если имеет место переход с 1 на 0, то выход Q равен 1. В противном случае он равен 0.

На размещение блока *Обнаружение отрицательного фронта сигнала* накладываются определенные ограничения (см. раздел 2.1).

Таблица 4–20. Блок "Обнаружение отрицательного фронта сигнала" и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	<адрес 1>	BOOL	I, Q, M, D, L	Сигнал, подлежащий контролю на появление отрицательного фронта
	M_BIT	BOOL	Q, M, D	Адрес M_BIT указывает бит памяти фронта, который хранит предыдущее состояние сигнала NEG. Используйте для M_BIT область памяти входов образа процесса (I) только в том случае, если этот адрес уже не занят ни одним из модулей ввода.
	Q	BOOL	I, Q, M, D, L	Выход с однократным импульсом

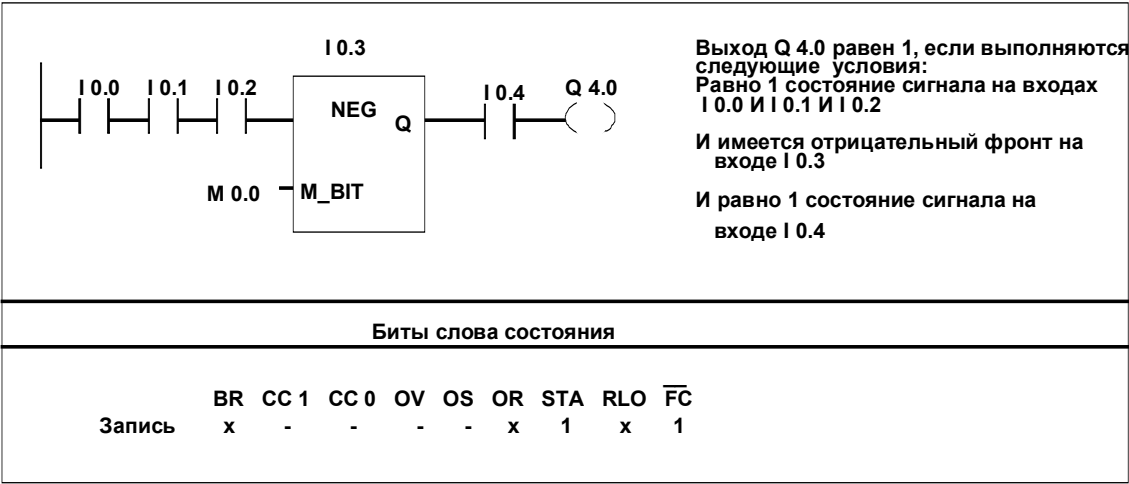


Рис. 4–20. Обнаружение отрицательного фронта сигнала

4.22 Установка–сброс триггера

Описание

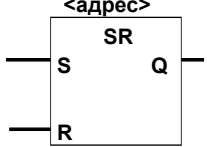
Команда *Установить–сбросить триггер (SR–триггер)* выполняет операции установки (S) и сброса (R) только тогда, когда RLO равен 1. RLO, равный 0, не оказывает влияния на эти операции; адрес, указанный в операции, остается неизменным.

SR–триггер устанавливается, если состояние сигнала равно 1 на входе S и равно 0 на входе R. В противном случае, если состояние сигнала равно 0 на входе S и 1 на входе R, триггер сбрасывается. Если RLO равен 1 на обоих входах, триггер сбрасывается.

Команда *Установить–сбросить триггер* испытывает воздействие со стороны *Главного управляющего реле (Master Control Relay, MCR)*. Дополнительную информацию о том, как функционирует MCR, можно получить в разделе 16.5.

На размещение блока *SR–триггер* накладываются определенные ограничения (см. раздел 2.1).

Таблица 4–21. Блок "Установка–сброс триггера" и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	<адрес>	BOOL	I, Q, M, D, L	Адрес указывает бит, который должен быть установлен или сброшен.
	S	BOOL	I, Q, M, D, L	Разрешенная операция установки
	R	BOOL	I, Q, M, D, L	Разрешенная операция сброса
	Q	BOOL	I, Q, M, D, L	Состояние сигнала <адрес>

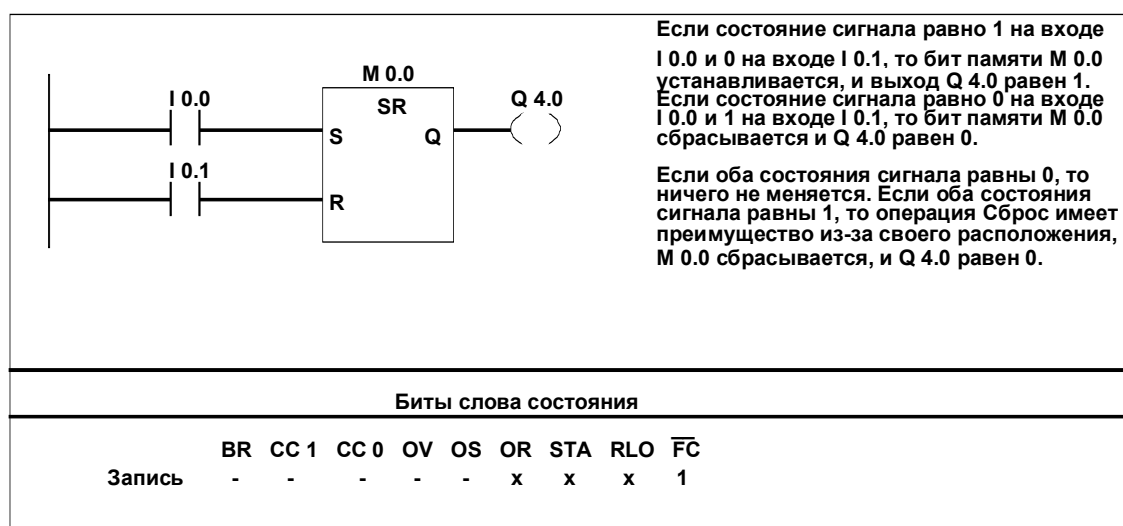


Рис. 4–21. Установка–сброс триггера

4.23 Сброс–установка триггера

Описание

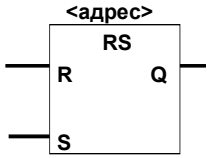
Команда *Сбросить–установить триггер (RS–триггер)* выполняет операции установки (S) и сброса (R) только тогда, когда RLO равен 1. RLO, равный 0, не оказывает влияния на эти операции; адрес, указанный в операции, остается неизменным.

RS–триггер сбрасывается, если состояние сигнала равно 1 на входе R и равно 0 на входе S. В противном случае, если состояние сигнала равно 0 на входе R и 1 на входе S, триггер устанавливается. set. Если RLO равен 1 на обоих входах, триггер устанавливается.

Команда *Сбросить–установить триггер* испытывает воздействие со стороны *Главного управляющего реле (Master Control Relay, MCR)*. Дополнительную информацию о том, как функционирует MCR, можно получить в разделе 16.5.

На размещение блока *RS–триггер* накладываются определенные ограничения (см. раздел 2.1).

Таблица 4–22. Блок "Сброс–установка триггера" и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	<адрес>	BOOL	I, Q, M, D, L	Адрес указывает бит, который должен быть установлен или сброшен.
	R	BOOL	I, Q, M, D, L	Разрешенная операция сброса
	S	BOOL	I, Q, M, D, L	Разрешенная операция установки
	Q	BOOL	I, Q, M, D, L	Состояние сигнала <адрес>

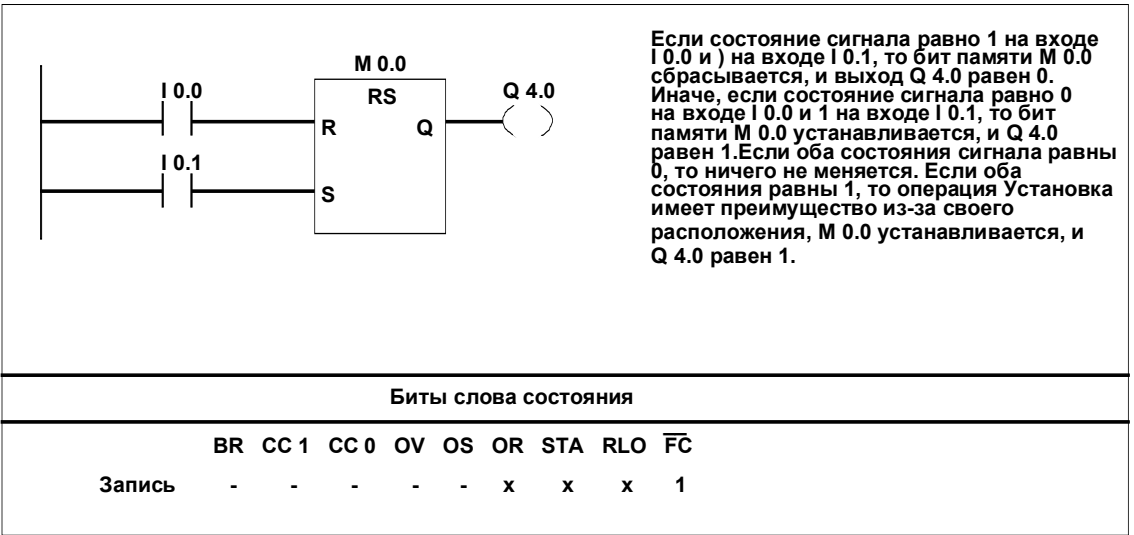


Рис. 4–22. Сброс–установка триггера

5 Таймерные команды

Обзор главы

Раздел	Описание	Page
5.1	Размещение таймера в памяти и его компоненты	5–2
5.2	Выбор подходящего таймера	5–4
5.3	Таймер S5 – формирователь импульса	5–5
5.4	Таймер S5 – формирователь удлиненного импульса	5–8
5.5	Таймер S5 – формирователь задержки включения	5–11
5.6	Таймер S5 – формирователь задержки включения с запоминанием	5–14
5.7	Таймер S5 – формирователь задержки выключения	5–17

5.1 Размещение таймера в памяти и его компоненты

Область в памяти

Таймеры имеют собственную зарезервированную область памяти в вашем CPU. Эта область памяти резервирует одно 16-битное слово для каждого адреса таймера. Набор команд контактного плана поддерживает 256 таймеров. Чтобы установить количество доступных таймерных слов, обратитесь к техническому описанию вашего.

К области памяти таймеров имеют доступ следующие функции:

- Таймерные команды
- Обновление таймерных слов с помощью генератора тактовых импульсов. Эта функция вашего CPU в режиме RUN уменьшает заданное значение времени на одну единицу через интервалы времени, установленные базой времени, пока значение времени не станет равным нулю.

Значение времени

Биты с 0 по 9 в таймерном слове содержат значение времени в двоичном коде. Значение времени задает количество единиц. Обновление таймера уменьшает значение времени на одну единицу через интервал времени, установленный базой времени. Значение времени уменьшается до тех пор, пока оно не станет равным нулю. Вы можете загружать значение времени в младшее слово аккумулятора 1 в двоичном, шестнадцатеричном или двоично-десятичном (BCD) коде (см. рис. 5–1). Диапазон времени охватывает значения с 0 по 9 990 секунд.

Вы можете предварительно загрузить значение времени с использованием любого из следующих форматов:

- W#16#wxyz
 - где w = база времени (то есть интервал времени или разрешающая способность)
 - xyz = значение времени в двоично-десятичном формате.
- S5T# aH_bbM_ccS_dddMS
 - где a = часы, bb = минуты, cc = секунды и ddd = миллисекунды
 - База времени выбирается автоматически и значение округляется до ближайшего меньшего числа с этой базой времени.

Максимальное значение времени, которое вы можете ввести, равно 9 990 секунд или 2H_46M_30S.

База времени

Биты 12 и 13 в таймерном слове содержат базу времени в двоичном коде. База времени определяет интервал, через который значение времени уменьшается на одну единицу (см. таблицу 5–1 и рис. 5–1). Минимальная база времени равна 10 мс; максимальная - 10 с.

Таблица 5–1. База времени и ее двоичный код

База времени	Двоичный код для базы времени
10 мс	00
100 мс	01
1 с	10
10 с	11

Так как значения времени запоминаются только через один интервал времени, то значения, не являющиеся точными кратными интервала времени, урезаются. Значения, разрешающая способность которых слишком велика для желаемого диапазона, округляются таким образом, что достигается желаемый диапазон, но не желаемая разрешающая способность. Таблица 5–2 показывает возможные разрешающие способности и соответствующие им диапазоны.

Таблица 5–2 Разрешающие способности и диапазоны для базы времени

Разрешающая способность	Диапазон
0,01 секунды	от 10MS до 9S_990MS
0,1 секунды	от 100MS до 1M_39S_900MS
1 секунда	от 1S до 16M_39S
10 секунд	от 10S до 2HR_46M_30S

Конфигурация битов в таймерной ячейке

Когда таймер запускается, содержимое таймерной ячейки используется в качестве значения времени. Биты с 0 по 11 в таймерной ячейке содержат значение времени в двоично-десятичном формате (BCD–формат: каждая группа из четырех битов содержит двоичный код одного десятичного разряда). Биты 12 и 13 содержат базу времени в двоичном коде (см. таблицу 5–1). Рис. 5–1 показывает содержимое таймерной ячейки, загруженной значением таймера 127 с базой времени 1 секунда.



Рис. 5–1. Содержимое таймерной ячейки для значения времени 127 и базы времени 1 секунда

Чтение времени и база времени

Каждый таймерный блок предоставляет два выхода, BI и BCD, для которых вы можете задать адрес слова. Значение времени на выходе BI представлено в двоичном формате. База времени и значение времени на выходе BCD представлены в двоично-десятичном формате (BCD).

5.2 Выбор подходящего таймера

Рис. 5–2 дает обзор пяти типов таймеров, описанных в данной главе. Этот обзор должен помочь вам выбрать таймер, адекватный вашим целям.

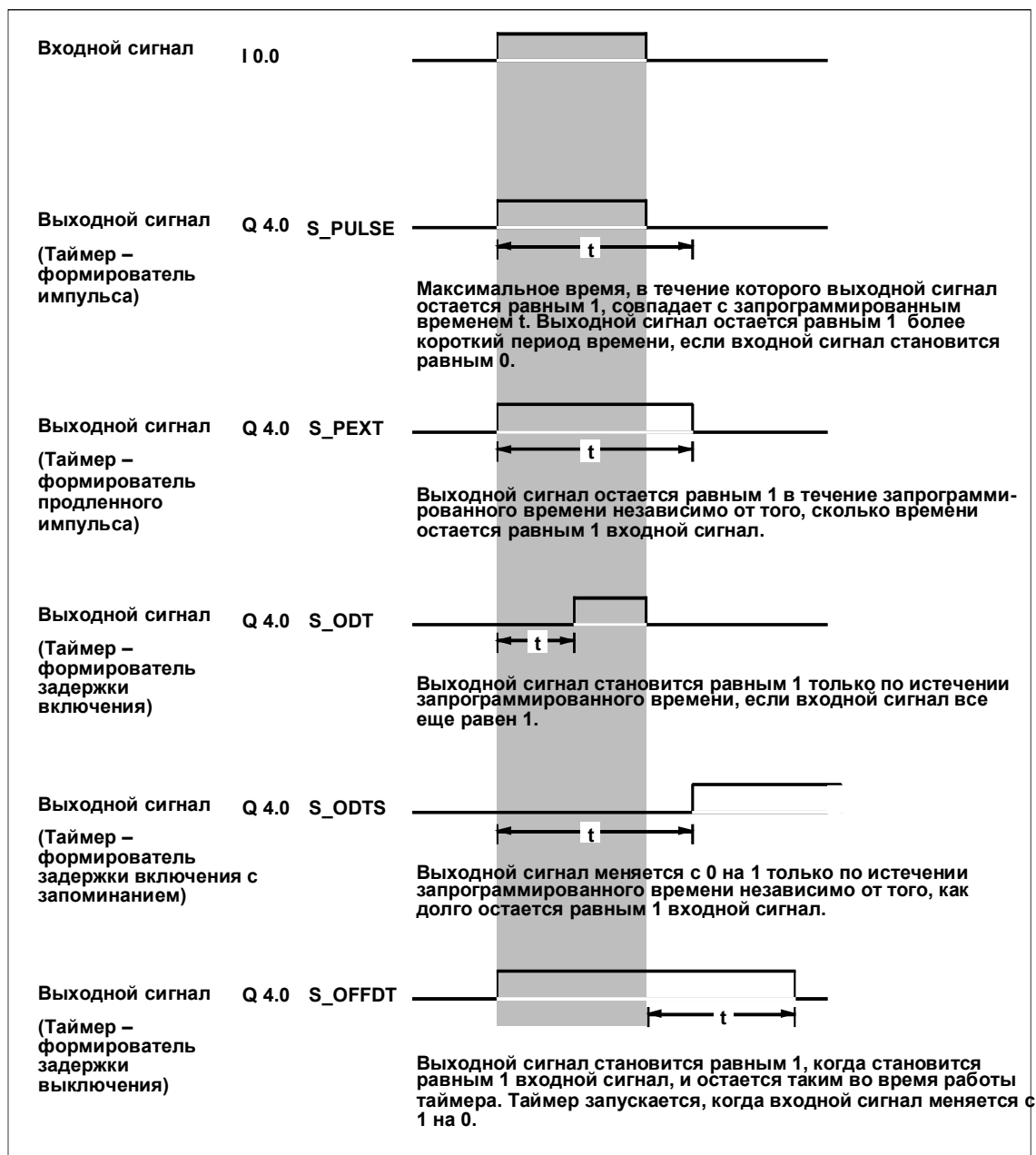


Рис. 5–2. Выбор подходящего таймера

5.3 Таймер S5 – формирователь импульса

Описание

Команда *Таймер S5 – формирователь импульса* запускает указанный таймер, если имеется положительный фронт (т.е. изменение состояния сигнала с 0 на 1) на входе S (Start [Пуск]). Изменение сигнала всегда необходимо для запуска таймера. Таймер продолжает работать с временем, указанным на входе TV (Time Value [Значение времени]), пока не истечет запрограммированное время, если состояние сигнала на входе TV равно 1. Пока таймер работает, опрос состояния сигнала на 1 на выходе Q дает результат, равный 1. Если на входе S происходит изменение с 1 на 0 до истечения заданного времени, таймер останавливается. Тогда опрос на 1 состояния сигнала на выходе Q дает результат, равный 0.

Изменение с 0 на 1 на входе таймера R (Reset [Сбросить]) во время работы таймера сбрасывает таймер. Это изменение сбрасывает также в ноль время и базу времени. Состояния сигнала 1 на входе таймера R не оказывает никакого влияния, если таймер не работает.

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на BI представлено в двоичном коде, а на выходе BCD – в двоично-десятичном коде.

Таблица 5–3. Блок "Таймер S5 – формирователь импульса" и параметры, с международным сокращенным именем

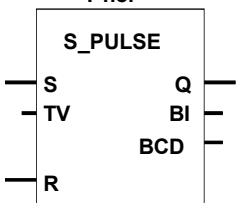
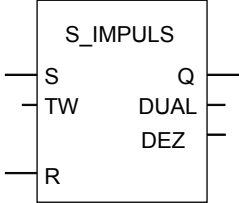
Блок КОР	Параметр	Тип данных	Область памяти	Описание
<p>T no.</p> 	no.	TIMER	T	Идентификационный номер таймера. Диапазон зависит от CPU.
	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
	TV	S5TIME	I, Q, M, D, L	Предустановленное значение времени (диапазон от 0 до 9999)
	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние таймера
	BI	WORD	I, Q, M, D, L	Остающееся время (целый формат)
	BCD	WORD	I, Q, M, D, L	Остающееся время (формат BCD)

Таблица 5–4. Блок "Таймер S5 – формирователь импульса" и параметры, с сокращенным именем SIMATIC

Блок КОР	Параметр	Тип данных	Область памяти	Описание
<p>T no.</p> 	no.	TIMER	T	Идентификационный номер таймера. Диапазон зависит от CPU.
	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
	TW	S5TIME	I, Q, M, D, L	Предустановленное значение времени (диапазон от 0 до 9999)
	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние таймера
	DUAL	WORD	I, Q, M, D, L	Остающееся время (целый формат)
	DEZ	WORD	I, Q, M, D, L	Остающееся время (формат BCD)

Пример

На рис. 5–3 показана команда *Таймер S5 – формирователь импульса*, описаны биты слова состояния и показаны характеристики таймера – формирователя импульса. На размещение таймерных блоков накладываются определенные ограничения (см. раздел 2.1).

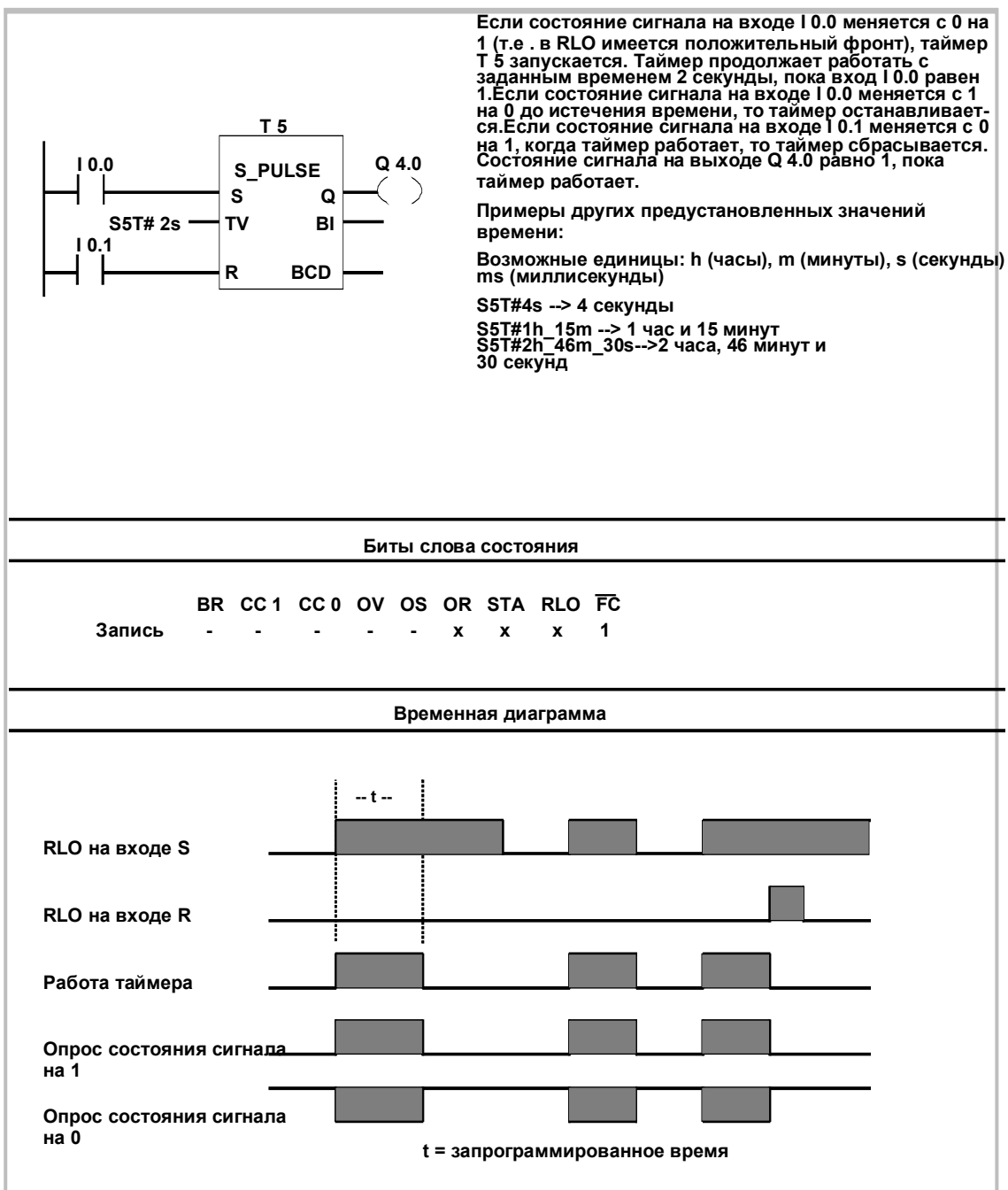


Рис. 5–3. Таймер S5 – формирователь импульса

5.4 Таймер S5 – формирователь удлиненного импульса

Описание

Команда *Таймер S5 – формирователь удлиненного импульса* запускает указанный таймер, если имеется положительный фронт (т.е. изменение состояния сигнала с 0 на 1) на входе S (Start [Пуск]). Изменение сигнала всегда необходимо для запуска таймера. Таймер продолжает работать с временем, указанным на входе TV (Time Value [Значение времени]), даже если состояние сигнала на входе S меняется на 0 до истечения времени. Пока таймер работает, опрос состояния сигнала на 1 на выходе Q дает результат, равный 1. Таймер перезапускается с заданным временем, если состояние сигнала на входе S меняется с 0 на 1 во время работы таймера. Изменение с 0 на 1 на входе таймера R (Reset [Сбросить]) во время работы таймера сбрасывает таймер. Это изменение сбрасывает также в ноль время и базу времени.

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на BI представлено в двоичном коде, а на выходе BCD – в двоично-десятичном коде.

Таблица 5–5. Блок "Таймер S5 – формирователь удлиненного импульса" и параметры, с международным сокращенным именем

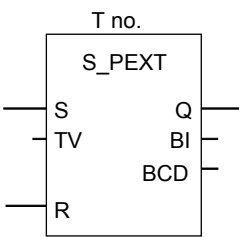
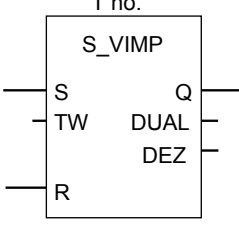
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	no.	TIMER	T	Идентификационный номер таймера. Диапазон зависит от CPU.
	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
	TV	S5TIME	I, Q, M, D, L	Предустановленное значение времени (диапазон от 0 до 9999)
	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние таймера
	BI	WORD	I, Q, M, D, L	Остающееся время (целый формат)
	BCD	WORD	I, Q, M, D, L	Остающееся время (формат BCD)

Таблица 5–6. Блок "Таймер S5 – формирователь удлиненного импульса" и параметры, с сокращенным именем SIMATIC

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	no.	TIMER	T	Идентификационный номер таймера. Диапазон зависит от CPU.
	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
	TW	S5TIME	I, Q, M, D, L	Предустановленное значение времени (диапазон от 0 до 9999)
	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние таймера
	DUAL	WORD	I, Q, M, D, L	Остающееся время (целый формат)
	DEZ	WORD	I, Q, M, D, L	Остающееся время (формат BCD)

Пример

На рис. 5–4 показана команда *Таймер S5 – формирователь удлиненного импульса*, описаны биты слова состояния и показаны характеристики таймера – формирователя удлиненного импульса. На размещение таймерных блоков накладываются определенные ограничения (см. раздел 2.1).

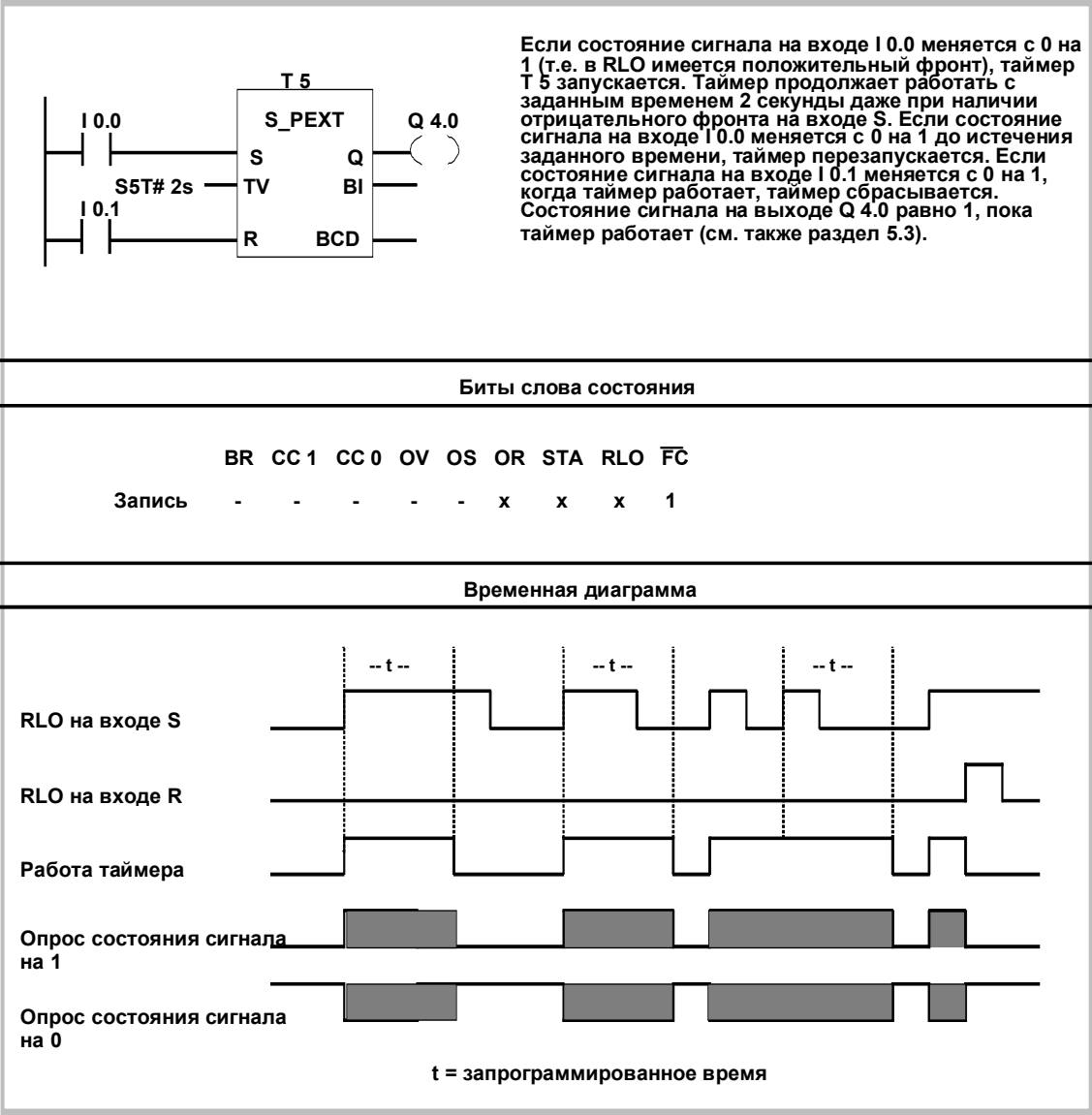


Рис. 5–4. Таймер S5 – формирователь удлиненного импульса

5.5 Таймер S5 – формирователь задержки включения

Описание

Команда *Таймер S5 – формирователь задержки включения* запускает указанный таймер, если имеется положительный фронт (т.е. изменение состояния сигнала с 0 на 1) на входе S (Start [Пуск]). Изменение сигнала всегда необходимо для запуска таймера. Таймер продолжает работать с временем, указанным на входе TV (Time Value [Значение времени]), пока состояние сигнала на входе S равно 1. Опрос состояния сигнала на 1 на выходе Q дает результат, равный 1, когда время истекло без ошибок, а состояние сигнала на входе S все еще равно 1. Когда во время работы таймера состояние сигнала на входе S меняется с 1 на 0, таймер останавливается. В этом случае опрос состояния сигнала на 1 на выходе Q всегда дает результат, равный 0.

Изменение с 0 на 1 на входе таймера R (Reset [Сбросить]) во время работы таймера сбрасывает таймер. Это изменение сбрасывает также в ноль время и базу времени. Таймер сбрасывается также, если состояние сигнала на входе R равно 1, когда таймер не работает.

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на BI представлено в двоичном коде, а на выходе BCD – в двоично-десятичном коде.

На размещение таймерных блоков накладываются определенные ограничения (см. раздел 2.1).

Таблица 5–7. Блок "Таймер S5 – формирователь задержки включения" и параметры, с международным сокращенным именем

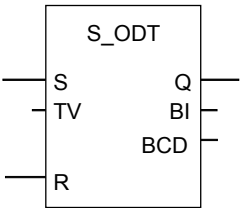
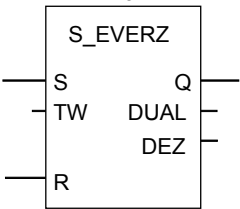
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	но.	TIMER	T	Идентификационный номер таймера. Диапазон зависит от CPU.
	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
	TV	S5TIME	I, Q, M, D, L	Предустановленное значение времени (диапазон от 0 до 9999)
	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние таймера
	BI	WORD	I, Q, M, D, L	Остающееся время (целый формат)
	BCD	WORD	I, Q, M, D, L	Остающееся время (формат BCD)

Таблица 5–8. Блок "Таймер S5 – формирователь задержки включения" и параметры, с сокращенным именем SIMATIC

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	no.	TIMER	T	Идентификационный номер таймера. Диапазон зависит от CPU.
	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
	TW	S5TIME	I, Q, M, D, L	Предустановленное значение времени (диапазон от 0 до 9999)
	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние таймера
	DUAL	WORD	I, Q, M, D, L	Остающееся время (целый формат)
	DEZ	WORD	I, Q, M, D, L	Остающееся время (формат BCD)

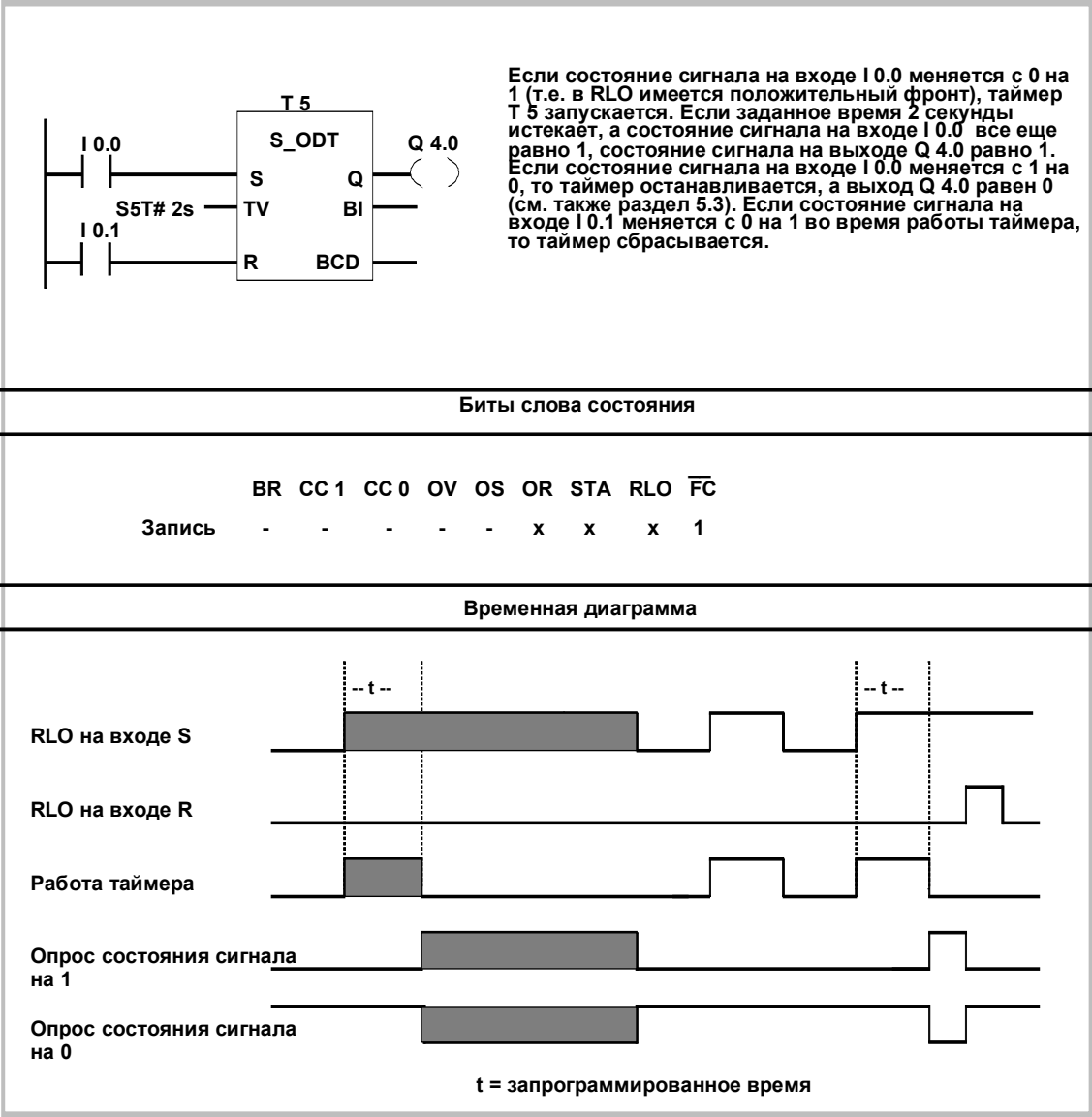


Рис. 5–5. Таймер S5 – формирователь задержки включения

5.6 Таймер S5 – формирователь задержки включения с запоминанием

Описание

Команда *Таймер S5 – формирователь задержки включения с запоминанием* запускает указанный таймер, если имеется положительный фронт (т.е. изменение состояния сигнала с 0 на 1) на входе S (Start [Пуск]). Изменение сигнала всегда необходимо для запуска таймера. Таймер продолжает работать с временем, указанным на входе TV (Time Value [Значение времени]), даже если состояние сигнала на входе S меняется на 0 до истечения заданного времени. Опрос состояния сигнала на 1 на выходе Q дает результат, равный 1, когда время истекло независимо от состояния сигнала на входе S, если вход сброса (R) остается равным 0. Таймер перезапускается с заданным временем, состояние сигнала на входе S меняется с 0 на 1 во время работы таймера.

Изменение с 0 на 1 на входе таймера R (Reset [Сбросить]) сбрасывает таймер независимо от RLO на входе S.

Таблица 5–9. Блок "Таймер S5 – формирователь задержки включения с запоминанием" и параметры, с международным сокращенным именем

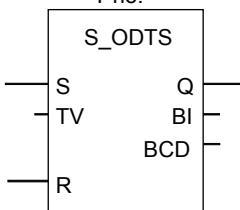
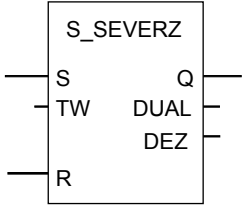
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	no.	TIMER	T	Идентификационный номер таймера. Диапазон зависит от CPU.
	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
	TV	S5TIME	I, Q, M, D, L	Предустановленное значение времени (диапазон от 0 до 9999)
	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние таймера
	BI	WORD	I, Q, M, D, L	Остающееся время (целый формат)
	BCD	WORD	I, Q, M, D, L	Остающееся время (формат BCD)

Таблица 5–10. Блок "Таймер S5 – формирователь задержки включения с запоминанием" и параметры, с сокращенным именем SIMATIC

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	no.	TIMER	T	Идентификационный номер таймера. Диапазон зависит от CPU.
	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
	TW	S5TIME	I, Q, M, D, L	Предустановленное значение времени (диапазон от 0 до 9999)
	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние таймера
	DUAL	WORD	I, Q, M, D, L	Остающееся время (целый формат)
	DEZ	WORD	I, Q, M, D, L	Остающееся время (формат BCD)

Пример

На рис. 5–6 показана команда *Таймер S5 – формирователь задержки включения с запоминанием*, описаны биты слова состояния и показаны характеристики таймера – формирователя задержки включения с запоминанием. На размещение таймерных блоков накладываются определенные ограничения (см. раздел 2.1).

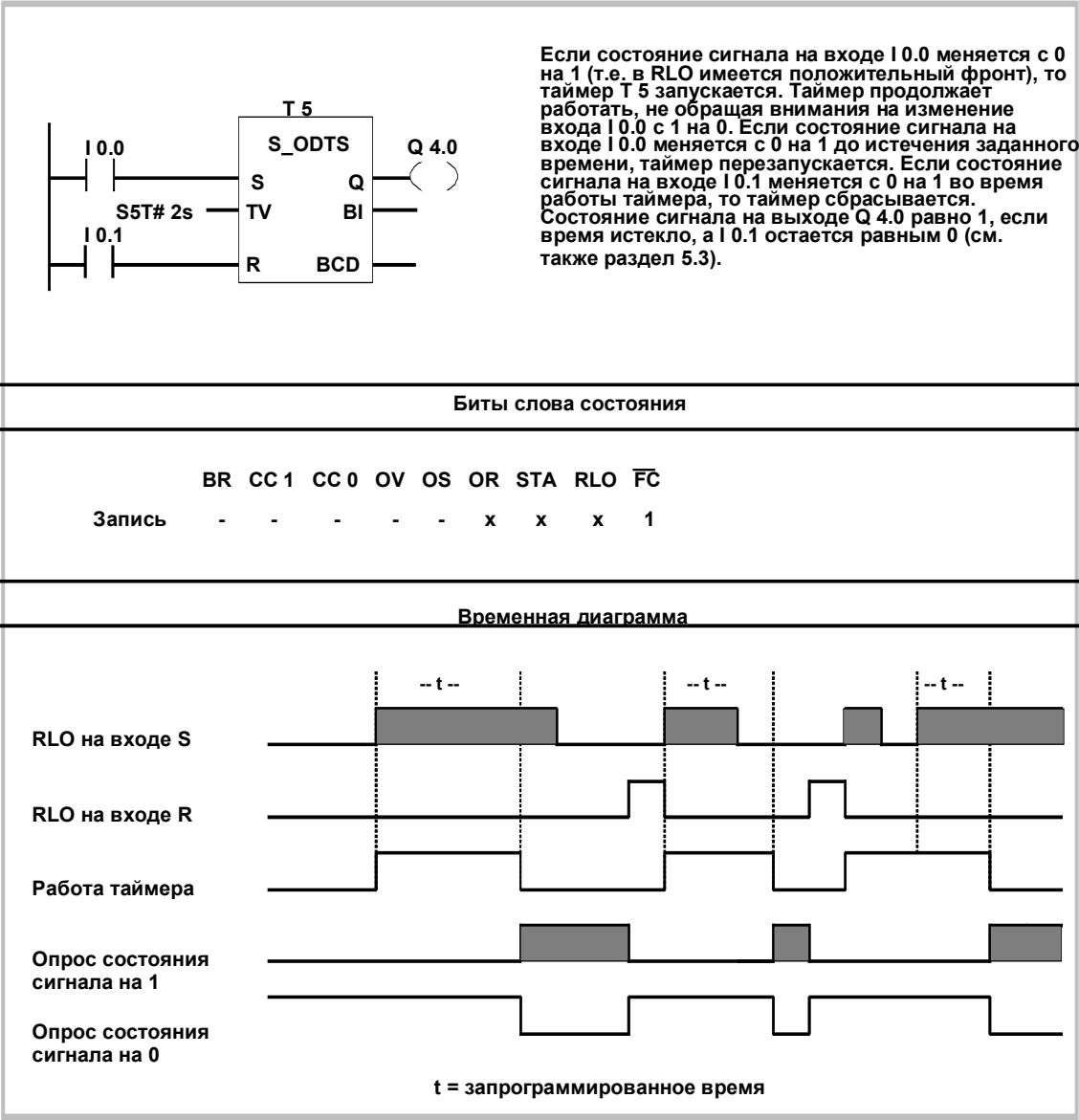


Рис. 5–6. Таймер S5 – формирователь задержки включения с запоминанием

5.7 Таймер S5 – формирователь задержки выключения

Описание

Команда *Таймер S5 – формирователь задержки выключения* запускает указанный таймер, если имеется отрицательный фронт (т.е. изменение состояния сигнала с 1 на 0) на входе S (Start [Пуск]). Изменение сигнала всегда необходимо для запуска таймера. Результат опроса состояния сигнала на 1 на выходе Q равен 1, когда равно 1 состояние сигнала на входе S или когда таймер работает. Таймер сбрасывается, когда состояние сигнала на входе S изменяется с 0 на 1 во время работы таймера. Таймер не перезапускается, пока состояние сигнала на входе S не изменится снова с 1 на 0.

Изменение с 0 на 1 на входе таймера R (Reset [Сбросить]) во время работы таймера сбрасывает таймер.

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на BI представлено в двоичном коде, а на выходе BCD – в двоично-десятичном коде.

На размещение таймерных блоков накладываются определенные ограничения (см. раздел 2.1).

Таблица 5–11. Блок "Таймер S5 – формирователь задержки выключения" и параметры, с международным сокращенным именем

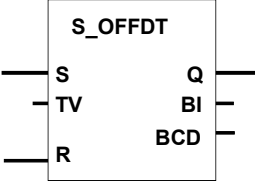
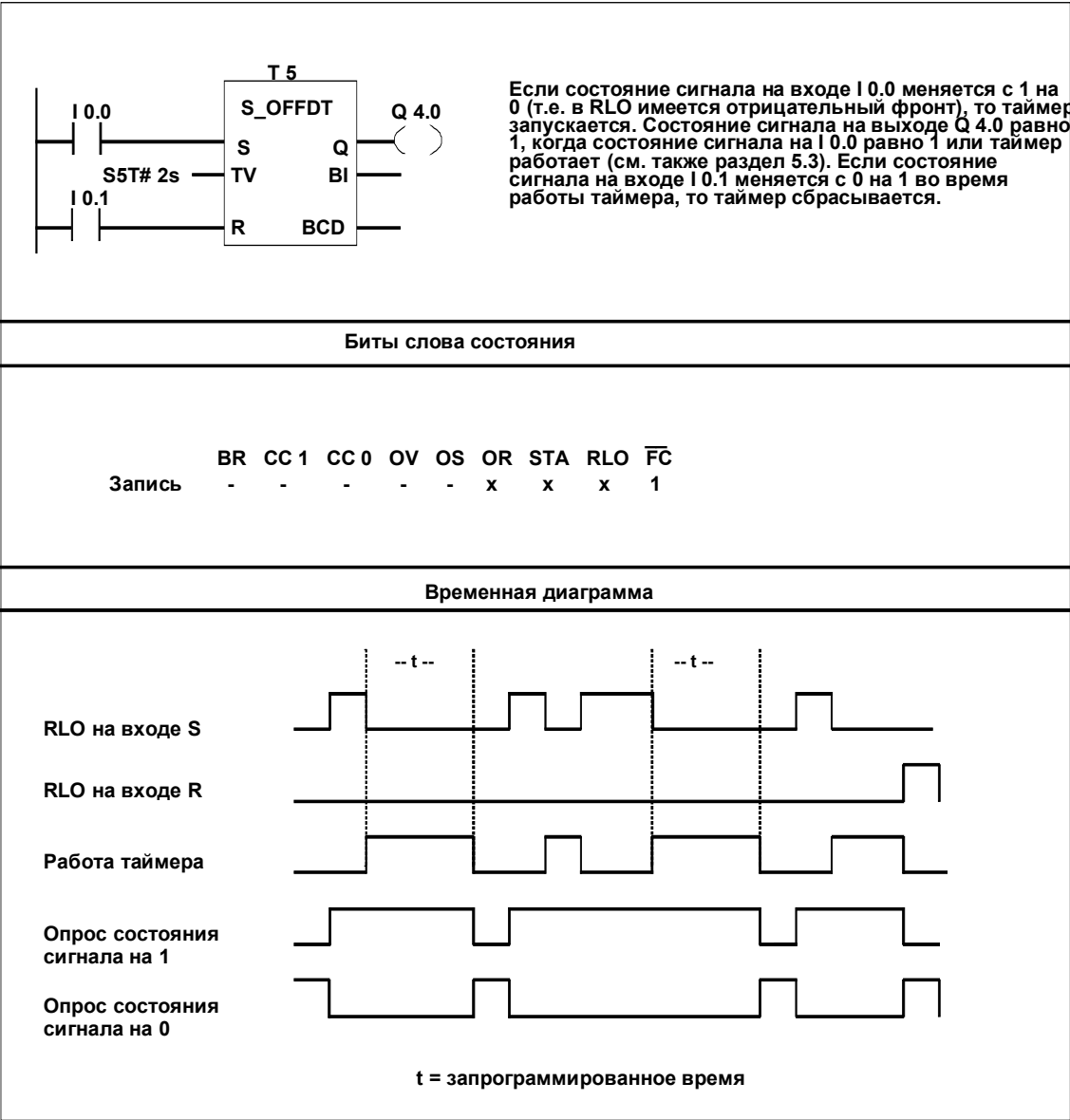
Блок КОР	Параметр	Тип данных	Область памяти	Описание
<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;">T no.</div>  </div>	no.	TIMER	T	Идентификационный номер таймера. Диапазон зависит от CPU.
	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
	TV	S5TIME	I, Q, M, D, L	Предустановленное значение времени (диапазон от 0 до 9999)
	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние таймера
	BI	WORD	I, Q, M, D, L	Остающееся время (целый формат)
	BCD	WORD	I, Q, M, D, L	Остающееся время (формат BCD)

Таблица 5–12. Блок "Таймер S5 – формирователь задержки выключения" и параметры, с сокращенным именем SIMATIC

Блок KOP	Параметр	Тип данных	Область памяти	Описание
<div style="text-align: center;"> <p>T no.</p> </div>	no.	TIMER	T	Идентификационный номер таймера. Диапазон зависит от CPU.
	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
	TW	S5TIME	I, Q, M, D, L	Предустановленное значение времени (диапазон от 0 до 9999)
	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние таймера
	DUAL	WORD	I, Q, M, D, L	Остающееся время (целый формат)
	DEZ	WORD	I, Q, M, D, L	Остающееся время (формат BCD)

Пример

На рис. 5–7 показана команда *Таймер S5 – формирователь задержки выключения*, описаны биты слова состояния и показаны характеристики таймера – формирователя задержки выключения.



6 Операции со счетчиками

Обзор главы

Раздел	Описание	Стр.
6.1	Размещение счетчика в памяти и его компоненты	6–2
6.2	Счетчик прямого и обратного счета	6–4
6.3	Счетчик прямого счета	6–6
6.4	Счетчик обратного счета	6–8

6.1 Размещение счетчика в памяти и его компоненты

Область в памяти

Счетчики имеют собственную зарезервированную область памяти в вашем CPU. Эта область памяти резервирует по одному 16-битному слову для каждого счетчика. Набор команд контактного плана поддерживает 256 счетчиков.

Операции счета являются единственными функциями, которые имеют доступ к области памяти, зарезервированной для счетчиков.

Счетное значение

Биты с 0 по 9 в слове счетчика содержат счетное значение в двоичном коде. Когда счетчик устанавливается, счетное значение передается из аккумулятора в слово счетчика. Диапазон счетных значений лежит между 0 и 999. Вы можете изменять счетное значение внутри этого диапазона, используя команды *Счетчик прямого и обратного счета*, *Счетчик прямого счета* и *Счетчик обратного счета*.

Конфигурация битов в счетчике

Вы устанавливаете счетчик на определенное значение, вводя число от 0 до 999, например, 127, в следующем формате:

C#127

C# означает двоично-десятичный формат (формат BCD: каждая группа из 4 битов содержит двоичный код одного десятичного разряда).

Биты счетчика с 0 по 11 содержат счетное значение в двоично-десятичном формате. На рис. 6–1 показано содержимое счетчика после того, как вы загрузили счетное значение 127, и содержимое ячейки счетчика после установки счетчика.



Рис. 6–1. Содержимое ячейки счетчика после того, как счетчик был установлен со счетным значением 127

6.2 Счетчик прямого и обратного счета

Описание

Положительный фронт (т.е. изменение состояния сигнала с 0 на 1) на входе S команды *Счетчик прямого и обратного счета* устанавливает счетчик значением на входе PV (Preset Value [Предустановленное значение]). Состояние сигнала 1 на входе R сбрасывает счетчик. Сброс счетчика делает счетное значение равным 0. Счетчик увеличивается на 1, если состояние сигнала на входе CU меняется с 0 на 1 (т.е. имеется положительный фронт), а значение счетчика меньше 999. Счетчик уменьшается на 1, если состояние сигнала на входе CD меняется с 0 на 1 (т.е. имеется положительный фронт), а значение счетчика больше 0. Если положительный фронт имеется на обоих входах, то обе операции выполняются и счетное значение остается тем же самым. Опрос состояния сигнала на 1 на выходе Q дает результат, равный 1, когда значение счетчика больше 0; опрос дает результат, равный 0, когда значение счетчика равно 0.

На размещение блоков счетчиков накладываются определенные ограничения (см. раздел 2.1).

Таблица 6–1. Блок "Счетчик прямого и обратного счета" и параметры, с международным сокращенным именем

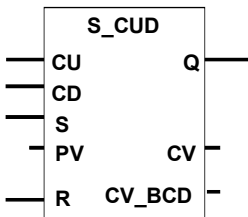
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	no.	COUNTER	C	Идентификационный номер счетчика. Диапазон зависит от CPU..
	CU	BOOL	I, Q, M, D, L	Вход прямого счета CU
	CD	BOOL	I, Q, M, D, L	Вход обратного счета CD
	S	BOOL	I, Q, M, D, L	Вход установки начального значения
	PV	WORD	I, Q, M, D, L	Значение в диапазоне от 0 до 999 для установки начального значения счетчика (вводится как C#<значение> для обозначения формата BCD)
	R	BOOL	I, Q, M, D, L	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние счетчика
	CV	WORD	I, Q, M, D, L	Текущее значение счетчика (целый формат)
	CV_BCD	WORD	I, Q, M, D, L	Текущее значение счетчика (формат BCD)

Таблица 6–2. Блок "Счетчик прямого и обратного счета" и параметры, с сокращенным именем SIMATIC

Блок KOP	Параметр	Тип данных	Область памяти	Описание
<div><div>Z no.</div><div><div>ZAHLER</div><div><div>ZV</div><div>ZR</div><div>S</div><div>ZW</div><div>R</div></div><div><div>Q</div><div>DUAL</div><div>DEZ</div></div></div></div>	no.	COUNTER	C	Идентификационный номер счетчика. Диапазон зависит от CPU.
	ZV	BOOL	I, Q, M, D, L	Вход прямого счета ZV
	ZR	BOOL	I, Q, M, D, L	Вход обратного счета ZR
	S	BOOL	I, Q, M, D, L	Вход установки начального значения
	ZW	WORD	I, Q, M, D, L	Значение в диапазоне от 0 до 999 для установки начального значения счетчика (вводится как C#<значение> для обозначения формата BCD)
	R	BOOL	I, Q, M, D, L	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние счетчика
	DUAL	WORD	I, Q, M, D, L	Текущее значение счетчика (целый формат)
	DEZ	WORD	I, Q, M, D, L	Текущее значение счетчика (формат BCD)

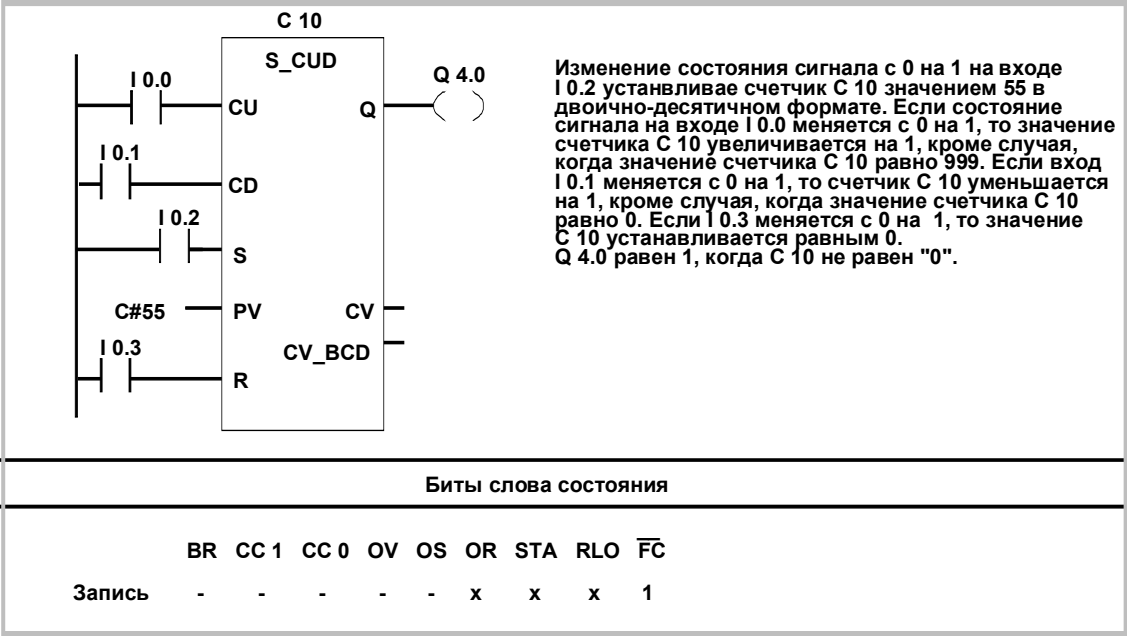


Рис. 6–2. Счетчик прямого и обратного счета

6.3 Счетчик прямого счета

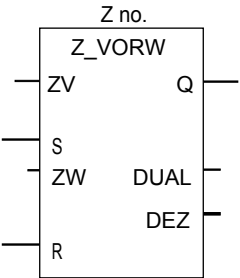
Описание

Положительный фронт (т.е. изменение состояния сигнала с 0 на 1) на входе S команды *Счетчик прямого счета* устанавливает счетчик значением на входе PV (Preset Value [Предустановленное значение]). Положительным фронтом на входе R счетчик сбрасывается. Сброс счетчика делает счетное значение равным 0. При положительном фронте на входе CU значение счетчика увеличивается на 1, если счетное значение меньше 999. Опрос состояния сигнала на 1 на выходе Q дает результат, равный 1, когда значение счетчика больше 0; опрос дает результат, равный 0, когда значение счетчика равно 0.

На размещение блоков счетчиков накладываются определенные ограничения (см. раздел 2.1).

Таблица 6–3. Блок "Счетчик прямого счета" и параметры, с международным сокращенным именем

Блок КОР	Параметр	Тип данных	Область памяти	Описание
<div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p style="text-align: center;">C no.</p> <p style="text-align: center;">S_CU</p> <div style="display: flex; justify-content: space-between;"> <div> <p>CU</p> <p>S</p> <p>PV</p> <p>R</p> </div> <div> <p>Q</p> <p>CV</p> <p>CV_BCD</p> </div> </div> </div>	no.	COUNTER	C	Идентификационный номер счетчика. Диапазон зависит от CPU.
	CU	BOOL	I, Q, M, D, L	Вход прямого счета CU
	S	BOOL	I, Q, M, D, L	Вход установки начального значения
	PV	WORD	I, Q, M, D, L	Значение в диапазоне от 0 до 999 для установки начального значения счетчика (вводится как C#<значение> для обозначения формата BCD)
	R	BOOL	I, Q, M, D, L	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние счетчика
	CV	WORD	I, Q, M, D, L	Текущее значение счетчика (целый формат)
	CV_BCD	WORD	I, Q, M, D, L	Текущее значение счетчика (формат BCD)

Таблица 6—4. Блок "Счетчик прямого счета" и параметры, с сокращенным именем SIMATIC				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	no.	COUNTER	C	Идентификационный номер счетчика. Диапазон зависит от CPU.
	ZV	BOOL	I, Q, M, D, L	Вход прямого счета ZV
	S	BOOL	I, Q, M, D, L	Вход установки начального значения
	ZW	WORD	I, Q, M, D, L	Значение в диапазоне от 0 до 999 для установки начального значения счетчика (вводится как C#<значение> для обозначения формата BCD)
	R	BOOL	I, Q, M, D, L	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние счетчика
	DUAL	WORD	I, Q, M, D, L	Текущее значение счетчика (целый формат)
	DEZ	WORD	I, Q, M, D, L	Текущее значение счетчика (формат BCD)

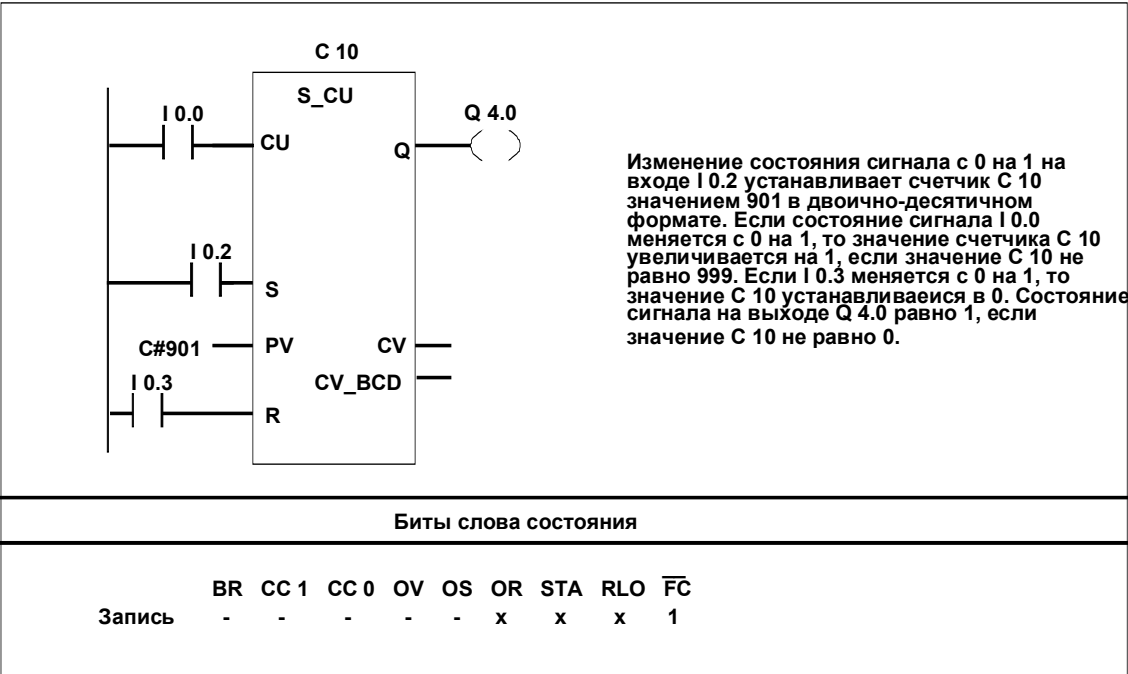


Рис. 6—3. Счетчик прямого счета

6.4 Счетчик обратного счета

Описание

Положительный фронт (т.е. изменение состояния сигнала с 0 на 1) на входе S команды *Счетчик обратного счета* устанавливает счетчик значением на входе PV (Preset Value [Предустановленное значение]). Положительным фронтом на входе R счетчик сбрасывается. Сброс счетчика делает счетное значение равным 0. При положительном фронте на входе CD значение счетчика уменьшается на 1, если счетное значение больше 0. Опрос состояния сигнала на 1 на выходе Q дает результат, равный 1, когда значение счетчика больше 0; опрос дает результат, равный 0, когда значение счетчика равно 0.

На размещение блоков счетчиков накладываются определенные ограничения (см. раздел 2.1).

Таблица 6–5. Блок "Счетчик обратного счета" и параметры, с международным сокращенным именем

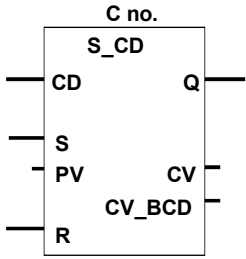
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	no.	COUNTER	C	Идентификационный номер счетчика. Диапазон зависит от CPU.
	CD	BOOL	I, Q, M, D, L	Вход обратного счета CD
	S	BOOL	I, Q, M, D, L	Вход установки начального значения
	PV	WORD	I, Q, M, D, L	Значение в диапазоне от 0 до 999 для установки начального значения счетчика (вводится как C#<значение> для обозначения формата BCD)
	R	BOOL	I, Q, M, D, L	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние счетчика
	CV	WORD	I, Q, M, D, L	Текущее значение счетчика (целый формат)
	CV_BCD	WORD	I, Q, M, D, L	Текущее значение счетчика (формат BCD)

Таблица 6–6. Блок "Счетчик обратного счета" и параметры, с сокращенным именем SIMATIC

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	no.	COUNTER	C	Идентификационный номер счетчика. Диапазон зависит от CPU.
	ZR	BOOL	I, Q, M, D, L	Вход обратного счета ZR
	S	BOOL	I, Q, M, D, L	Вход установки начального значения
	ZW	WORD	I, Q, M, D, L	Значение в диапазоне от 0 до 999 для установки начального значения счетчика (вводится как C#<значение> для обозначения формата BCD)
	R	BOOL	I, Q, M, D, L	Вход сброса
	Q	BOOL	I, Q, M, D, L	Состояние счетчика
	DUAL	WORD	I, Q, M, D, L	Текущее значение счетчика (целый формат)
	DEZ	WORD	I, Q, M, D, L	Текущее значение счетчика (формат BCD)

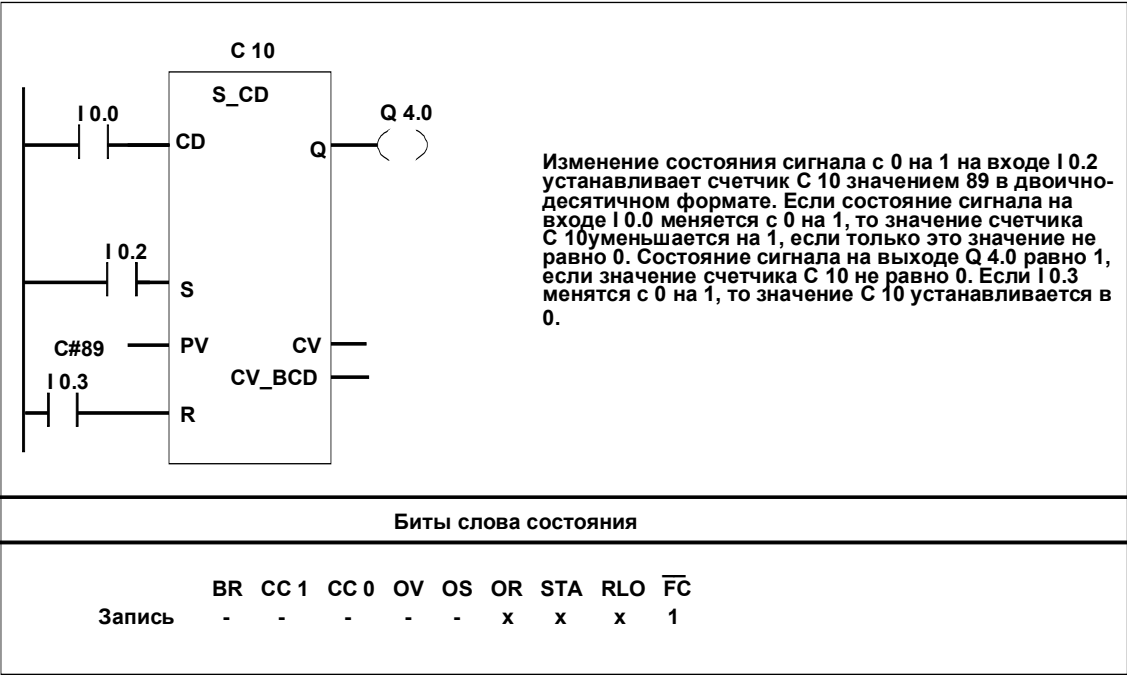


Рис. 6–4. Счетчик обратного счета

7 Операции с целыми числами

Обзор главы

Раздел	Описание	Стр.
7.1	Сложение целых чисел	7–2
7.2	Сложение двойных целых чисел	7–3
7.3	Вычитание целых чисел	7–4
7.4	Вычитание двойных целых чисел	7–5
7.5	Умножение целых чисел	7–6
7.6	Умножение двойных целых чисел	7–7
7.7	Деление целых чисел	7–8
7.8	Деление двойных целых чисел	7–9
7.9	Получение остатка от деления двойного целого числа	7–10
7.10	Оценка битов слова состояния после операций с целыми числами	7–11

7.1 Сложение целых чисел

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Сложение целых чисел*. Эта команда складывает входы IN1 и IN2. Результат может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для целых чисел, то биты OV и OS слова состояния имеют значение 1, а ENO - значение 0.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 7–1. Блок "Сложение целых чисел" и параметры				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	INT	I, Q, M, D, L	Первое слагаемое
	IN2	INT	I, Q, M, D, L	Второе слагаемое
	OUT	INT	I, Q, M, D, L	Сумма

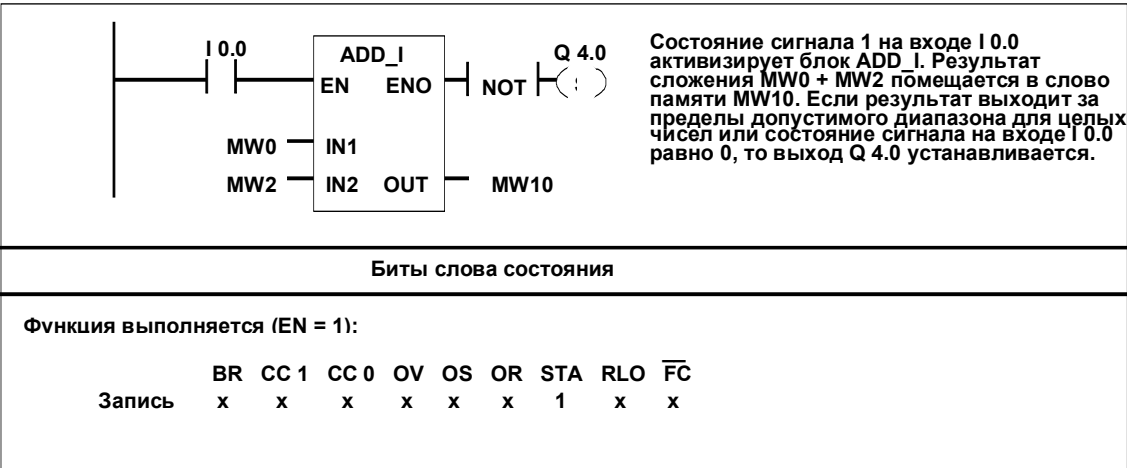


Рис. 7–1. Сложение целых чисел

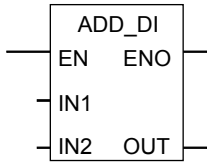
7.2 Сложение двойных целых чисел

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Сложение двойных целых чисел*. Эта команда складывает входы IN1 и IN2. Результат может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для двойных целых чисел, то биты OV и OS слова состояния имеют значение 1, а ENO - значение 0.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 7–2. Блок "Сложение двойных целых чисел" и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	DINT	I, Q, M, D, L	Первое слагаемое
	IN2	DINT	I, Q, M, D, L	Второе слагаемое
	OUT	DINT	I, Q, M, D, L	Сумма

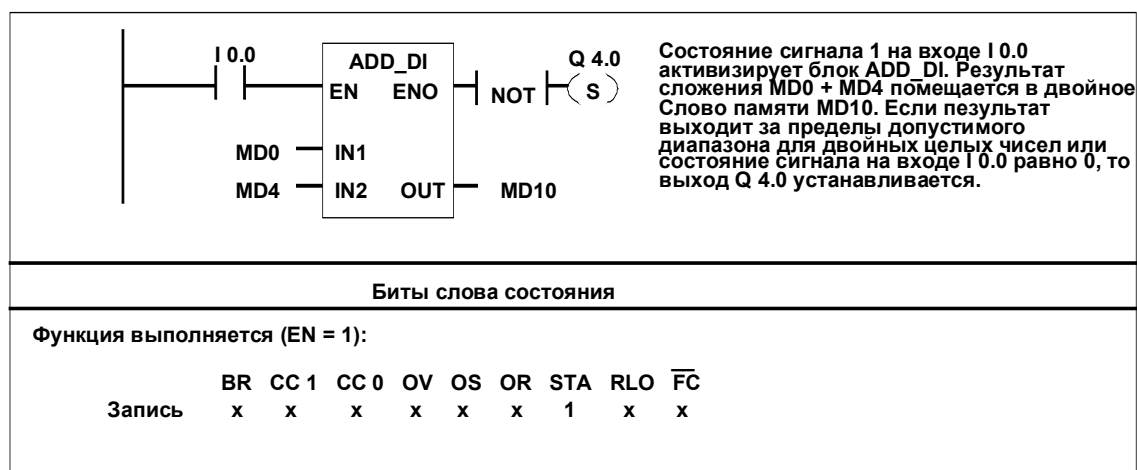


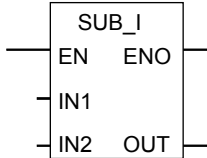
Рис. 7–2. Сложение двойных целых чисел

7.3 Вычитание целых чисел

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Вычитание целых чисел*. Эта команда вычитает вход IN2 из IN1. Результат может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для целых чисел, то биты OV и OS имеют значение 1, а ENO - значение 0.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 7–3. Блок "Вычитание целых чисел" и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	INT	I, Q, M, D, L	Уменьшаемое
	IN2	INT	I, Q, M, D, L	Вычитаемое
	OUT	INT	I, Q, M, D, L	Разность



Состояние сигнала 1 на входе I 0.0 активизирует блок SUB_I. Результат вычитания MW0 - MW2 помещается в слово памяти MW10. Если результат выходит за пределы допустимого диапазона для целых чисел или состояние сигнала на входе I 0.0 равно 0, то выход Q 4.0 устанавливается.

Биты слова состояния									
Функция выполняется (EN = 1):									
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	x	x	x	x	x	1	x	x

Рис. 7–3. Вычитание целых чисел

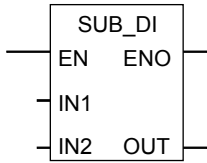
7.4 Вычитание двойных целых чисел

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Вычитание двойных целых чисел*. Эта команда вычитает вход IN2 из IN1. Результат может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для двойных целых чисел, то биты OV и OS имеют значение 1, а ENO - значение 0.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 7–4. Блок "Вычитание двойных целых чисел" и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	DINT	I, Q, M, D, L	Уменьшаемое
	IN2	DINT	I, Q, M, D, L	Вычитаемое
	OUT	DINT	I, Q, M, D, L	Разность

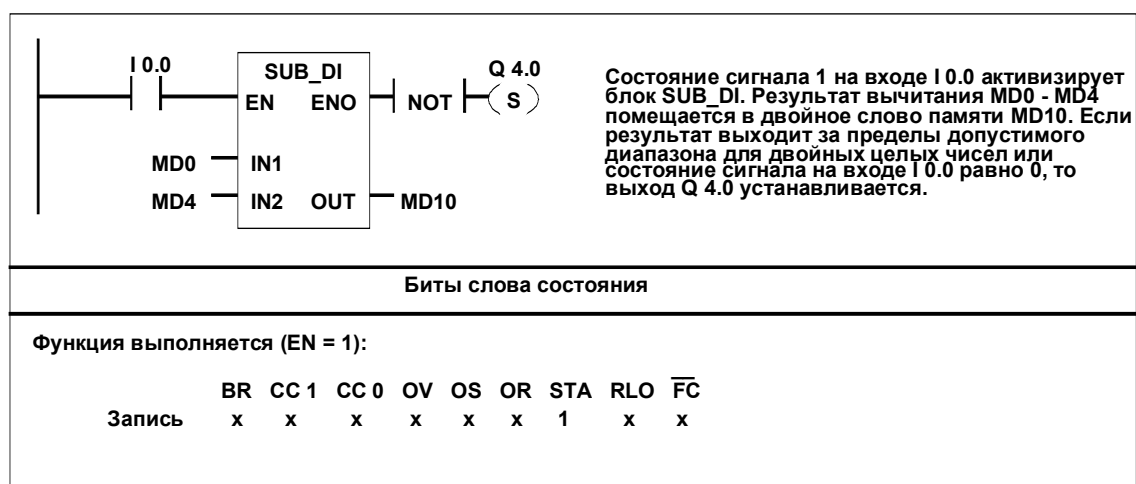


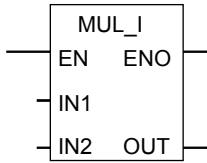
Рис. 7–4. Вычитание двойных целых чисел

7.5 Умножение целых чисел

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Умножение целых чисел*. Эта команда умножает вход IN1 на IN2. Результат может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для целых чисел, то биты OV и OS имеют значение 1, а ENO - значение 0.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 7–5. Блок "Умножение целых чисел" и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	INT	I, Q, M, D, L	Первый сомножитель
	IN2	INT	I, Q, M, D, L	Второй сомножитель
	OUT	DINT	I, Q, M, D, L	Произведение



Состояние сигнала 1 на входе I 0.0 активизирует блок MUL_I. Результат умножения MW0 x MW2 помещается в двойное слово памяти MD10. Если результат выходит за пределы допустимого диапазона для 16-битных целых чисел или состояние сигнала на входе I 0.0 равно 0, то выход Q 4.0 устанавливается.

Биты слова состояния

Функция выполняется (EN = 1):

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	x	x	x	x	x	1	x	x

Рис. 7–5. Умножение целых чисел

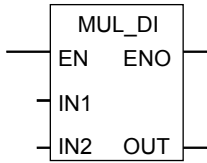
7.6 Умножение двойных целых чисел

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Умножение двойных целых чисел*. Эта команда умножает вход IN1 на IN2. Результат может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для двойных целых чисел, то биты OV и OS имеют значение 1, а ENO - значение 0.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 7–6. Блок "Умножение двойных целых чисел" и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	DINT	I, Q, M, D, L	Первый сомножитель
	IN2	DINT	I, Q, M, D, L	Второй сомножитель
	OUT	DINT	I, Q, M, D, L	Произведение

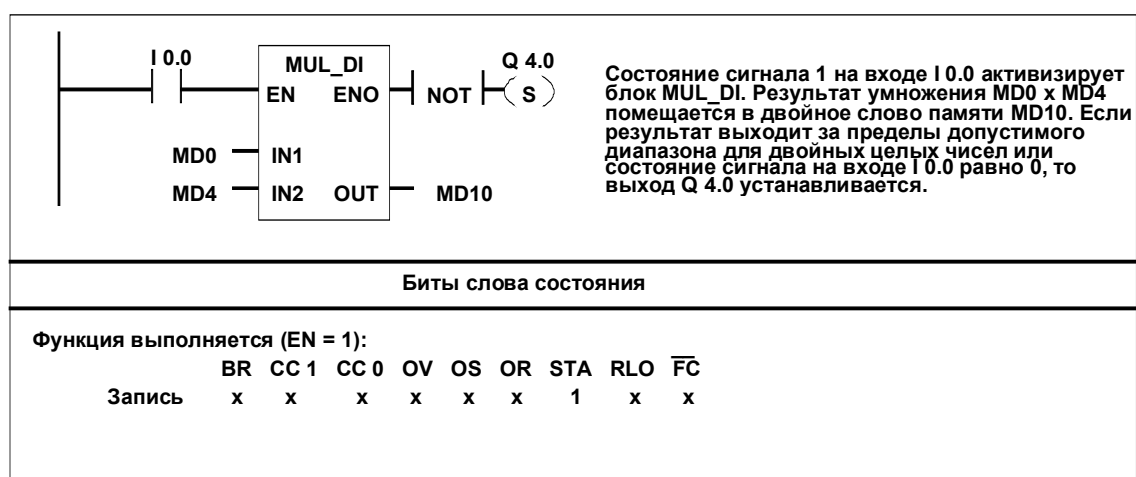


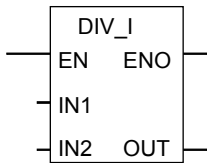
Рис. 7–6. Умножение двойных целых чисел

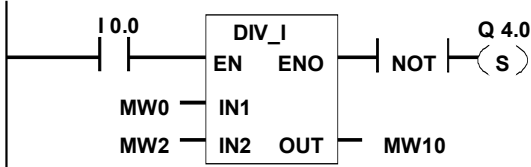
7.7 Деление целых чисел

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Деление целых чисел*. Эта команда делит вход IN1 на IN2. Частное от этого деления (округленный до целого результат) может быть опрошен на выходе OUT. Остаток от деления не может опрашиваться. Если частное лежит вне допустимого диапазона для целых чисел, то биты OV и OS имеют значение 1, а ENO - значение 0.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 7–7. Блок "Деление целых чисел" и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	INT	I, Q, M, D, L	Делимое
	IN2	INT	I, Q, M, D, L	Делитель
	OUT	INT	I, Q, M, D, L	Частное



Состояние сигнала 1 на входе I 0.0 активизирует блок DIV_I. Частное от деления MW0 на MW2 помещается в слово памяти MW10. Если частное выходит за пределы допустимого диапазона для целых чисел или состояние сигнала на входе I 0.0 равно 0, то выход Q 4.0 устанавливается.

Биты слова состояния

Функция выполняется (EN = 1):

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	x	x	x	x	x	1	x	x

Рис. 7–7. Деление целых чисел

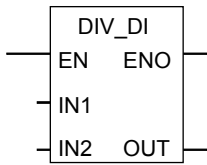
7.8 Деление двойных целых чисел

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Деление двойных целых чисел*. Эта команда делит вход IN1 на IN2. Частное от этого деления (округленный до целого результат) может быть опрошен на выходе OUT. Команда *Деление двойных целых чисел* сохраняет частное в виде одного 32-битного значения в формате DINT и не формирует остатка от деления. Если частное лежит вне допустимого диапазона для двойных целых чисел, то биты OV и OS имеют значение 1, а ENO - значение 0.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 7–8. Блок "Деление двойных целых чисел" и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	DINT	I, Q, M, D, L	Делимое
	IN2	DINT	I, Q, M, D, L	Делитель
	OUT	DINT	I, Q, M, D, L	Частное

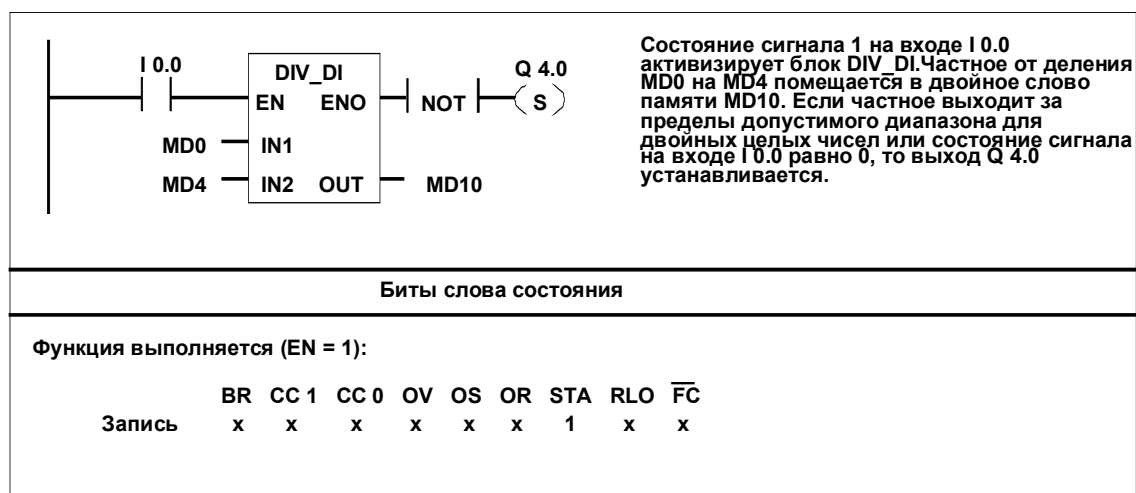


Рис. 7–8. Деление двойных целых чисел

7.9 Получение остатка от деления двойного целого числа

Описание

Состояние сигнала "1" на разрешающем входе (EN) активизирует команду *Получение остатка от деления двойного целого числа*. Эта операция делит вход IN1 на IN2. Остаток от деления может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для двойных целых чисел, то биты OV и OS имеют значение 1, а ENO - значение 0.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 7–9. Блок "Получение остатка от деления двойного целого числа" и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	DINT	I, Q, M, D, L	Делимое
	IN2	DINT	I, Q, M, D, L	Делитель
	OUT	DINT	I, Q, M, D, L	Остаток

Состояние сигнала 1 на входе I 0.0 активизирует блок MOD. Остаток от деления MD0 на MD4 хранится в двойном слове памяти MD10. Если результат выходит за пределы допустимого диапазона для двойных целых чисел или состояние сигнала на входе I 0.0 равно выход Q 4.0 устанавливается.

Биты слова состояния									
Функция выполняется (EN = 1):									
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	x	x	x	x	x	1	x	x

Рис. 7–9. Получение остатка от деления двойного целого числа

7.10 Оценка битов слова состояния после операций с целыми числами

Основные математические команды оказывают воздействие на следующие биты слова состояния:

- CC 1 и CC 0
- OV
- OS

Тире (–) в таблице означает, что результат операции не оказывает влияния на этот бит.

Таблица 7–10. Состояние сигнала битов слова состояния: результат в допустимом диапазоне				
Допустимый диапазон для результата операции с целыми числами (16 и 32 бита)	Биты слова состояния			
	CC 1	CC 0	OV	OS
0 (ноль)	0	0	0	–
16 битов: $-32\,768 \leq \text{результат} < 0$ (отрицательное число) 32 бита: $-2\,147\,483\,648 \leq \text{результат} < 0$ (отрицательное число)	0	1	0	–
16 битов: $32\,767 \geq \text{результат} > 0$ (положительное число) 32 бита: $2\,147\,483\,647 \geq \text{результат} > 0$ (положительное число)	1	0	0	–

Таблица 7–11. Состояние сигнала битов слова состояния: результат вне допустимого диапазона				
Недопустимый диапазон для результата операции с целыми числами (16 и 32 бита)	Биты слова состояния			
	CC 1	CC 0	OV	OS
16 битов: результат $> 32\,767$ (положительное число) 32 бита: результат $> 2\,147\,483\,647$ (положительное число)	1	0	1	1
16 битов: результат $< -32\,768$ (отрицательное число) 32 бита: результат $< -2\,147\,483\,648$ (отрицательное число)	0	1	1	1

Таблица 7–12. Состояние сигнала битов слова состояния: операции с целыми числами (32 бита) +D, /D и MOD				
Команда	Биты слова состояния			
	CC 1	CC 0	OV	OS
+D: результат = $-4\,294\,967\,296$	0	0	1	1
/D или MOD: деление на 0	1	1	1	1

8 Операции над числами с плавающей точкой

Обзор главы

Раздел	Описание	Стр.
8.1	Обзор	8–2
8.2	Сложение чисел с плавающей точкой	8–3
8.3	Вычитание чисел с плавающей точкой	8–4
8.4	Умножение чисел с плавающей точкой	8–5
8.5	Деление чисел с плавающей точкой	8–6
8.6	Оценка битов слова состояния после выполнения операций с плавающей точкой	8–7
8.7	Образование абсолютного значения числа с плавающей точкой	8–8
8.8	Образование квадрата и/или квадратного корня числа с плавающей точкой	8–9
8.9	Образование натурального логарифма числа с плавающей точкой	8–11
8.10	Образование экспоненциального значения числа с плавающей точкой	8–12
8.11	Образование тригонометрических функций углов в виде чисел с плавающей точкой	8–13

8.1 Обзор

Вы можете использовать команды арифметики с плавающей точкой для выполнения следующих операций над двумя 32-битными числами IEEE с плавающей точкой:

- сложение
- вычитание
- умножение
- деление

32-битные числа с плавающей точкой IEEE относятся к типу данных REAL. Используя арифметику с плавающей точкой, вы можете выполнять следующие операции с **одним** 32-битным числом IEEE с плавающей точкой:

- образование квадрата (SQR) и квадратного корня (SQRT) числа с плавающей точкой
- образование натурального логарифма (LN) числа с плавающей точкой
- образование экспоненциального значения (EXP) с основанием e ($= 2.71828\dots$)
- образование следующих тригонометрических функций угла, представленного в виде 32-битного числа IEEE с плавающей точкой:
 - образование синуса (SIN) числа с плавающей точкой и образование арксинуса (ASIN) числа с плавающей точкой
 - образование косинуса (COS) числа с плавающей точкой и образование арккосинуса (ACOS) числа с плавающей точкой
 - образование тангенса (TAN) числа с плавающей точкой и образование арктангенса (ATAN) числа с плавающей точкой

8.2 Сложение чисел с плавающей точкой

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Сложение чисел с плавающей точкой*. Эта команда складывает входы IN1 и IN2. Результат может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для чисел с плавающей точкой (переполнение или потеря значимости), то биты OV и OS слова состояния имеют значение 1, а ENO - значение 0. Информацию по оценке этих индикаторов в слове состояния вы найдете в разделе 8.6.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 8–1. Блок "Сложение вещественных чисел" и параметры				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	REAL	I, Q, M, D, L	Первое слагаемое
	IN2	REAL	I, Q, M, D, L	Второе слагаемое
	OUT	REAL	I, Q, M, D, L	Сумма

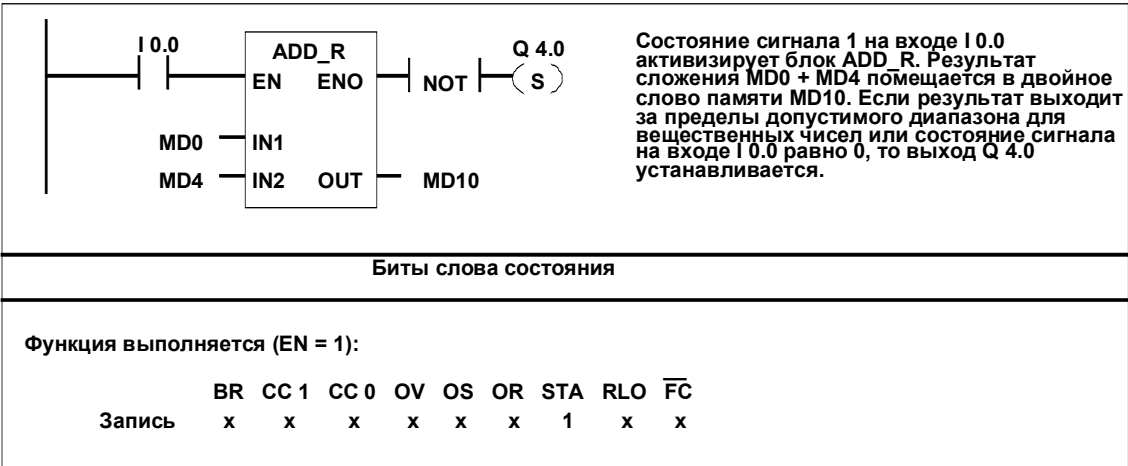


Рис. 8–1. Сложение вещественных чисел

8.3 Вычитание чисел с плавающей точкой

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Вычитание чисел с плавающей точкой*. Эта команда вычитает вход IN2 из IN1. Результат может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для чисел с плавающей точкой (переполнение или потеря значимости), то биты OV и OS слова состояния имеют значение 1, а ENO - значение 0. Информацию по оценке этих индикаторов в слове состояния вы найдете в разделе 8.6.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 8–2. Блок "Вычитание вещественных чисел" и параметры				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	REAL	I, Q, M, D, L	Уменьшаемое
	IN2	REAL	I, Q, M, D, L	Вычитаемое
	OUT	REAL	I, Q, M, D, L	Разность



Рис. 8–2. Вычитание вещественных чисел

8.4 Умножение чисел с плавающей точкой

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Умножение чисел с плавающей точкой*. Эта команда умножает вход IN1 на IN2. Результат может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для чисел с плавающей точкой (переполнение или потеря значимости), то биты OV и OS слова состояния имеют значение 1, а ENO - значение 0. Информацию по оценке этих индикаторов в слове состояния вы найдете в разделе 8.6.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 8–3. Блок "Умножение вещественных чисел" и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	REAL	I, Q, M, D, L	Первый сомножитель
	IN2	REAL	I, Q, M, D, L	Второй сомножитель
	OUT	REAL	I, Q, M, D, L	Произведение

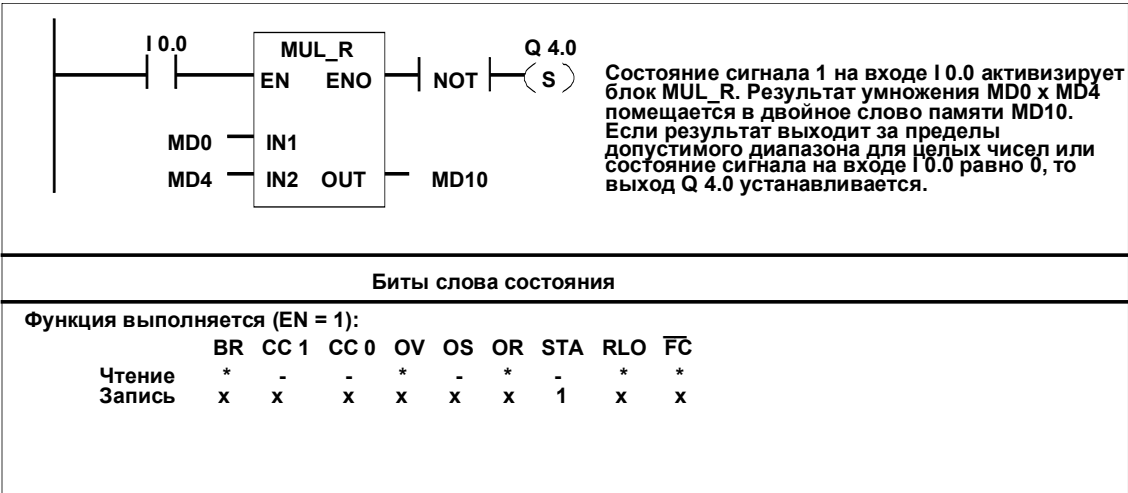


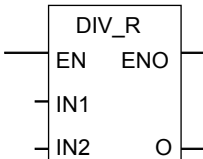
Рис. 8–3. Умножение вещественных чисел

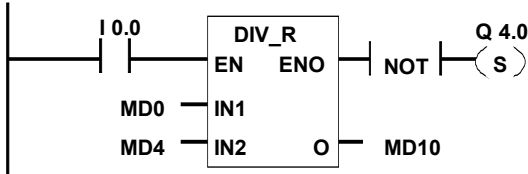
8.5 Деление чисел с плавающей точкой

Описание

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Деление чисел с плавающей точкой*. Эта команда делит вход IN1 на IN2. Результат может быть опрошен на выходе OUT. Если результат лежит вне допустимого диапазона для чисел с плавающей точкой (переполнение или потеря значимости), то биты OV и OS слова состояния имеют значение 1, а ENO - значение 0. Информацию по оценке этих индикаторов в слове состояния вы найдете в разделе 8.6.

На размещение блоков операций с целыми числами накладываются определенные ограничения (см. раздел 2.1).

Таблица 8–4. Блок "Деление вещественных чисел" и параметры				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	REAL	I, Q, M, D, L	Делимое
	IN2	REAL	I, Q, M, D, L	Делитель
	O	REAL	I, Q, M, D, L	Частное



Состояние сигнала 1 на входе I 0.0 активизирует блок DIV_R. Результат деления MD0 на MD4 помещается в двойное слово памяти MD10. Если результат выходит за пределы допустимого диапазона для вещественных чисел или состояние сигнала на входе I 0.0 равно 0, то выход Q 4.0 устанавливается.

Биты слова состояния

Функция выполняется (EN = 1):

BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	x	x	x	x	1	x	x

Рис. 8–4. Деление вещественных чисел

8.6 Оценка битов слова состояния после выполнения операций с плавающей точкой

Описание

Математические операции оказывают влияние на следующие биты слова состояния:

- CC 1 и CC 0
- OV
- OS

Тире (-) в столбце битов таблицы означает, что соответствующий бит не испытывает воздействия со стороны результата операции.

Таблица 8–5. Состояние сигнала битов слова состояния для результатов операций с плавающей точкой, находящихся в допустимом диапазоне значений				
Допустимый диапазон для результата операции над числами с плавающей точкой (32 бита)	Биты слова состояния			
	CC 1	CC 0	OV	OS
+0, –0 (ноль)	0	0	0	-
$-3.402823\text{E}+38 < \text{результат} < -1.175494\text{E}-38$ (отрицательное число)	0	1	0	-
$+1.175494\text{E}-38 < \text{результат} < 3.402823\text{E}+38$ (положительное число)	1	0	0	-

Таблица 8–6. Состояние сигнала битов слова состояния для результатов операций с плавающей точкой, находящихся вне допустимого диапазона значений				
Недопустимый диапазон для результата операции над числами с плавающей точкой (32 бита)	Биты слова состояния			
	CC 1	CC 0	OV	OS
$-1.175494\text{E}-38 < \text{результат} < -1.401298\text{E}-45$ (отрицательное число) потеря значимости	0	0	1	1
$+1.401298\text{E}-45 < \text{результат} < +1.175494\text{E}-38$ (положительное число) потеря значимости	0	0	1	1
результат $< -3.402823\text{E}+38$ (отрицательное число) переполнение	0	1	1	1
результат $> 3.402823\text{E}+38$ (положительное число) переполнение	1	0	1	1
результат $< -3.402823\text{E}+38$ или результат $> +3.402823\text{E}+38$ не является числом с плавающей точкой	1	1	1	1

8.7 Образование абсолютного значения числа с плавающей точкой

Описание

С помощью команды *Образование абсолютного значения числа с плавающей точкой* вы можете получить абсолютное значение числа с плавающей точкой.

Таблица 8–7. Блок ABS и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Входная величина: вещественное число
	OUT	REAL	I, Q, M, D, L	Выходная величина: абсолютное значение вещественного числа



Рис. 8–5. Образование абсолютного значения числа с плавающей точкой

8.8 Образование квадрата и/или квадратного корня числа с плавающей точкой

Описание

С помощью команды *Образование квадрата числа с плавающей точкой* вы можете возвести в квадрат число с плавающей точкой.

С помощью команды *Образование квадратного корня числа с плавающей точкой* вы можете извлечь квадратный корень из числа с плавающей точкой. Эта команда выводит положительный результат, если операнд больше, чем "0". Единственное исключение: квадратный корень из - 0 равен -0.

Информацию о воздействии, оказываемом командами SQR и SQRT на состояние сигнала битов состояния CC 1, CC 0, OV и OS, вы найдете в разделе 8.6.

Параметры

Таблица 8–8 показывает блок SQR и описывает его параметры. Таблица 8–9 показывает блок SQRT и описывает его параметры.

Таблица 8–8. Блок SQR и параметры

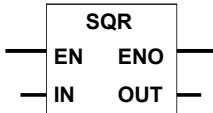
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Число
	OUT	REAL	I, Q, M, D, L	Квадрат числа

Таблица 8–9. Блок SQRT и параметры

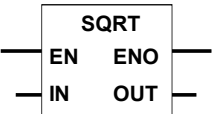
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Число
	OUT	REAL	I, Q, M, D, L	Квадратный корень из числа



Рис. 8–6. Образование квадратного корня из числа с плавающей точкой

8.9 Образование натурального логарифма числа с плавающей точкой

Описание

С помощью команды *Образование натурального логарифма числа с плавающей точкой* вы можете образовать натуральный логарифм числа с плавающей точкой.

Информацию о воздействии, оказываемом командой LN на биты состояния CC 1, CC 0, OV и OS, вы найдете в разделе 8.6.

Таблица 8–10. Блок LN и параметры				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
<div><div>LN</div><div>EN ENO</div><div>IN OUT</div></div>	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Число
	OUT	REAL	I, Q, M, D, L	Натуральный логарифм числа

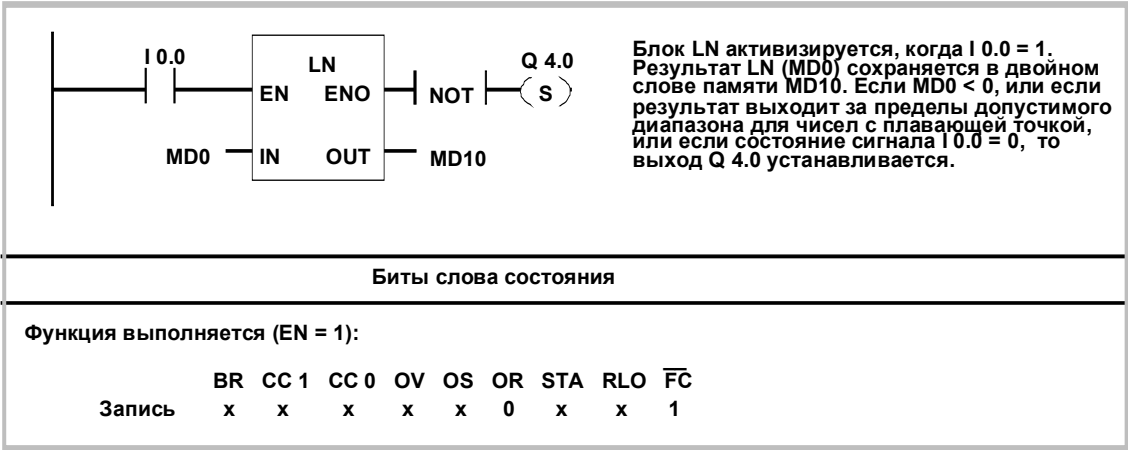


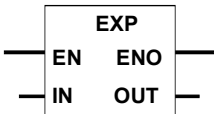
Рис. 8–7. Образование натурального логарифма числа с плавающей точкой

8.10 Образование экспоненциального значения числа с плавающей точкой

Описание

С помощью команды *Образование экспоненциального значения числа с плавающей точкой* вы можете образовать экспоненциальное значение числа с плавающей точкой с основанием e ($= 2.71828\dots$).

Информацию о воздействии, оказываемом командой EXP на биты состояния CC 1, CC 0, OV и OS, вы найдете в разделе 8.6.

Таблица 8–11. Блок EXP и параметры				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Число
	OUT	REAL	I, Q, M, D, L	Экспонента числа

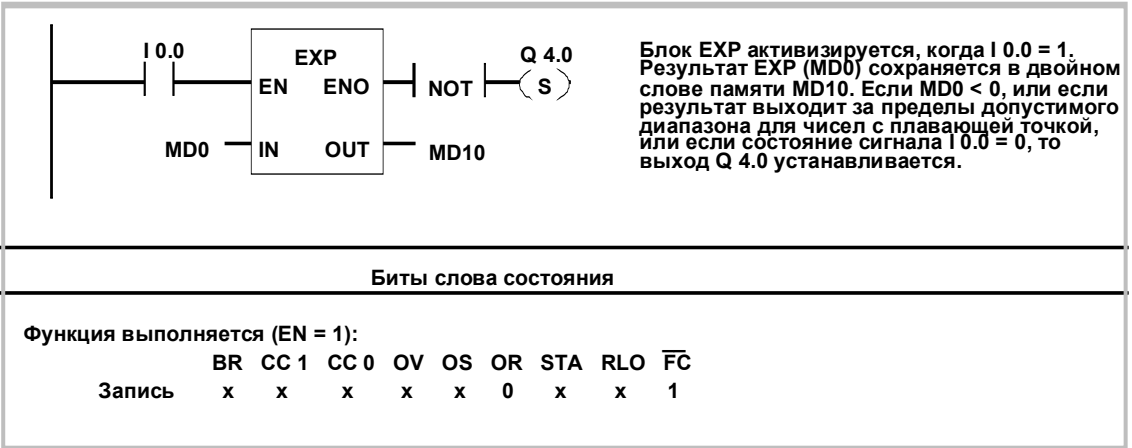


Рис. 8–8. Образование экспоненциального значения числа с плавающей точкой

8.11 Образование тригонометрических функций углов в виде чисел с плавающей точкой

Описание

С помощью следующих команд вы можете образовать тригонометрические функции углов, представленных в виде чисел с плавающей точкой (32 бита, IEEE).

Команда	Объяснение
SIN	Образовать синус угла, заданного в радианах.
ASIN	Образовать арксинус числа с плавающей точкой. Результатом является угол, заданный в радианах. Это значение находится в следующем диапазоне: $-\pi / 2 \leq \text{арксинус} \leq +\pi / 2$, где $\pi = 3.14...$
COS	Образовать косинус числа с плавающей точкой, представляющего угол, заданный в радианах.
ACOS	Образовать арккосинус числа с плавающей точкой. Результатом является угол, заданный в радианах. Это значение находится в следующем диапазоне: $0 \leq \text{арккосинус} \leq +\pi$, где $\pi = 3.14...$
TAN	Образовать тангенс числа с плавающей точкой, представляющего угол, заданный в радианах.
ATAN	Образовать арктангенс числа с плавающей точкой. Результатом является угол, заданный в радианах. Это значение находится в следующем диапазоне: $-\pi / 2 \leq \text{арктангенс} \leq +\pi / 2$, где $\pi = 3.14...$

Информацию о воздействии, оказываемом командами SIN, ASIN, COS, ACOS, TAN и ATAN на биты состояния CC 1, CC 0, OV и OS, вы найдете в разделе 8.6.

Параметры

Таблицы 8–12 - 8–17 показывают блоки SIN, ASIN, COS, ACOS, TAN и ATAN и описывают их параметры.

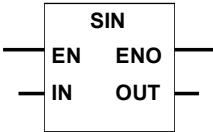
Таблица 8–12. Блок SIN и параметры				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Число
	OUT	REAL	I, Q, M, D, L	Синус числа

Таблица 8–13. Блок ASIN и параметры

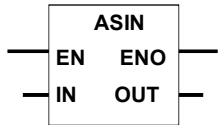
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Число
	OUT	REAL	I, Q, M, D, L	Арксинус числа

Таблица 8–14. Блок COS и параметры

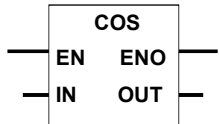
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Число
	OUT	REAL	I, Q, M, D, L	Косинус числа

Таблица 8–15. Блок ACOS и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Число
	OUT	REAL	I, Q, M, D, L	Арккосинус числа

Таблица 8–16. Блок TAN и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Число
	OUT	REAL	I, Q, M, D, L	Тангенс числа

Таблица 8–17. Блок ATAN и параметры

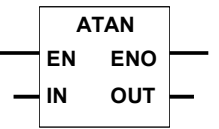
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Число
	OUT	REAL	I, Q, M, D, L	Арктангенс числа



Рис. 8–9. Образование синуса числа с плавающей точкой

9 Команды сравнения

Обзор главы

Раздел	Описание	Стр.
9.1	Сравнение целых чисел	9–2
9.2	Сравнение двойных целых чисел	9–3
9.3	Сравнение чисел с плавающей точкой	9–5

9.1 Сравнение целых чисел

Описание

Команда *Сравнение целых чисел* выполняет операцию сравнения над 16-битными числами с фиксированной точкой. Вы можете использовать эту команду как обычный контакт. Эта команда сравнивает входы IN1 и IN2 в соответствии с видом сравнения, который вы выбираете в браузере. В таблице 9–1 перечислены допустимые сравнения.

Если сравнение истинно, то результат логической операции (RLO) этой функции равен 1. В противном случае он равен 0. Отрицание выхода сравнения отсутствует, так как этот результат может быть получен применением соответствующей обратной функции сравнения.

Таблица 9–1. Типы сравнения для целых чисел	
Тип сравнения	Символы в названии в верхней части блока
IN1 равен IN2.	==
IN1 не равен IN2.	<>
IN1 больше, чем IN2.	>
IN1 меньше, чем IN2.	<
IN1 больше или равен IN2.	>=
IN1 меньше или равен IN2.	<=

Таблица 9–2. Блок "Сравнение целых чисел" и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	IN1	INT	I, Q, M, D, L	Первое сравниваемое значение
	IN2	INT	I, Q, M, D, L	Второе сравниваемое значение

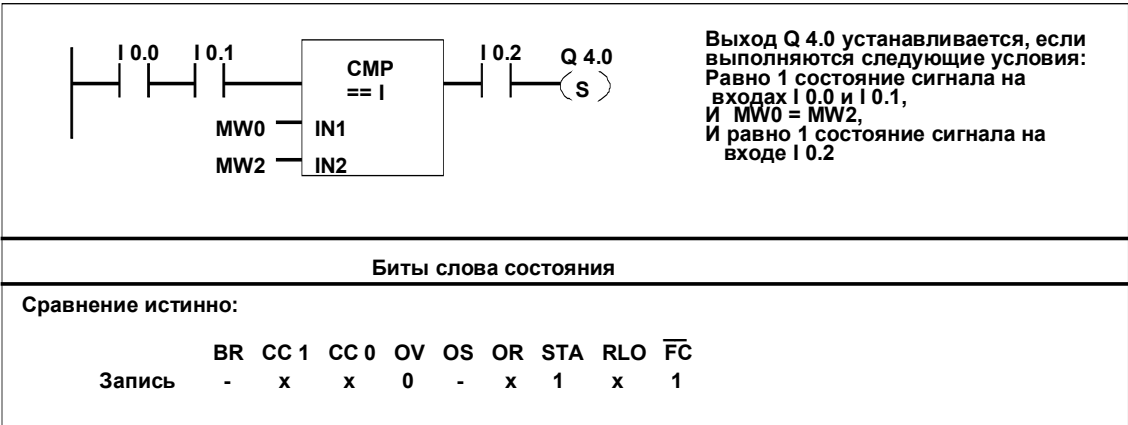


Рис. 9–1. Сравнение целых чисел

9.2 Сравнение двойных целых чисел

Описание

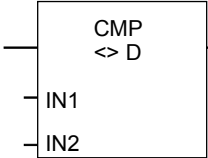
Команда *Сравнение двойных целых чисел* выполняет операцию сравнения над 32-битными числами с фиксированной точкой. Вы можете использовать эту команду как обычный контакт. Эта команда сравнивает входы IN1 и IN2 в соответствии с видом сравнения, который вы выбираете в браузере. В таблице 9–3 перечислены допустимые сравнения.

Если сравнение истинно, то результат логической операции (RLO) этой функции равен 1. В противном случае он равен 0. Отрицание выхода сравнения отсутствует, так как этот результат может быть получен применением соответствующей обратной функции сравнения.

Таблица 9–3. Типы сравнения для двойных целых чисел

Тип сравнения	Символы в названии в верхней части блока
IN1 равен IN2.	==
IN1 не равен IN2.	<>
IN1 больше, чем IN2.	>
IN1 меньше, чем IN2.	<
IN1 больше или равен IN2.	>=
IN1 меньше или равен IN2.	<=

Таблица 9–4. Блок "Сравнение двойных целых чисел" и параметры (пример: не равно)

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	IN1	DINT	I, Q, M, D, L	Первое сравниваемое значение
	IN2	DINT	I, Q, M, D, L	Второе сравниваемое значение

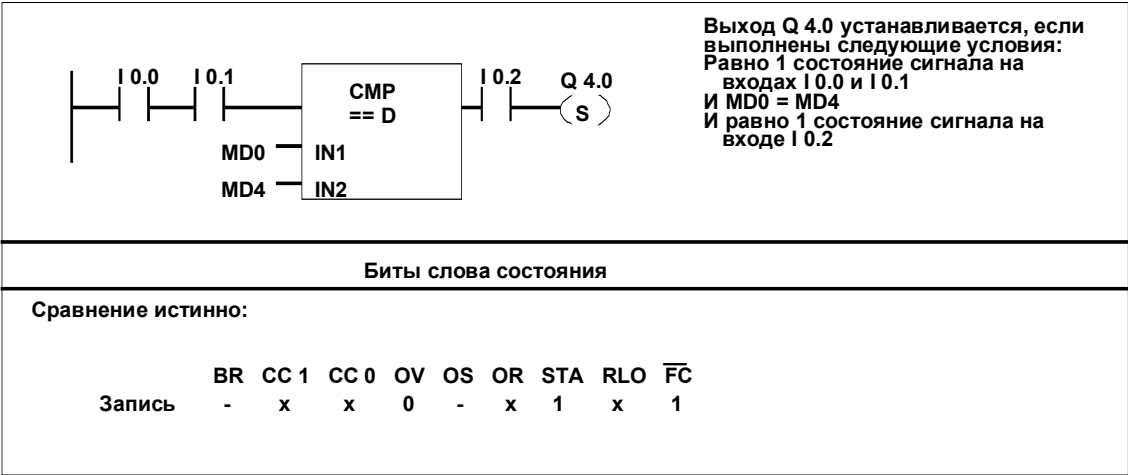


Рис. 9–2. Сравнение двойных целых чисел

9.3 Сравнение чисел с плавающей точкой

Описание

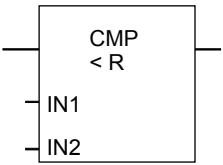
Команда *Сравнение чисел с плавающей точкой* запускает операцию сравнения. Вы можете использовать эту команду как обычный контакт. Эта команда сравнивает входы IN1 и IN2 в соответствии с видом сравнения, который вы выбираете в браузере. В таблице 9–5 перечислены допустимые сравнения.

Если сравнение истинно, то результат логической операции (RLO) этой функции равен 1. В противном случае он равен 0. Отрицание выхода сравнения отсутствует, так как этот результат может быть получен применением соответствующей обратной функции сравнения.

Таблица 9–5. Типы сравнения для чисел с плавающей точкой

Тип сравнения	Символы в названии в верхней части блока
IN1 равен IN2.	==
IN1 не равен IN2.	<>
IN1 больше, чем IN2.	>
IN1 меньше, чем IN2.	<
IN1 больше или равен IN2.	>=
IN1 меньше или равен IN2.	<=

Таблица 9–6. Сравнение чисел с плавающей точкой: блок и параметры (пример: меньше, чем)

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	IN1	REAL	I, Q, M, D, L	Первое сравниваемое значение
	IN2	REAL	I, Q, M, D, L	Второе сравниваемое значение

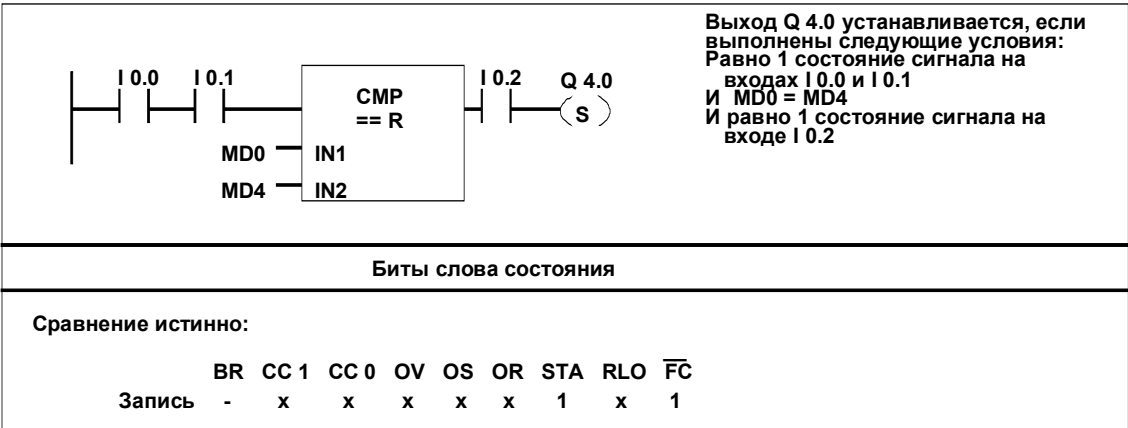


Рис. 9–3. Сравнение чисел с плавающей точкой

10 Команды пересылки и преобразования

Обзор главы

Раздел	Описание	Страница
10.1	Присваивание значения	10–2
10.2	Преобразование двоично-десятичного числа в целое	10–4
10.3	Преобразование целого числа в двоично-десятичное	10–5
10.4	Преобразование целого числа в двойное целое	10–6
10.5	Преобразование двоично-десятичного числа в двойное целое	10–7
10.6	Преобразование двойного целого числа в двоично-десятичное	10–8
10.7	Преобразование двойного целого числа в число с плавающей точкой	10–9
10.8	Дополнение целого числа до единицы	10–10
10.9	Дополнение двойного целого числа до единицы	10–11
10.10	Дополнение целого числа до двух	10–12
10.11	Дополнение двойного целого числа до двух	10–13
10.12	Изменение знака числа с плавающей точкой	10–14
10.13	Округление до двойного целого числа	10–15
10.14	Выделение целой части числа	10–16
10.15	Округление до ближайшего большего целого числа	10–17
10.16	Округление до ближайшего меньшего целого числа	10–18

10.1 Присваивание значения

Описание

Команда *Присвоить значение* дает возможность предварительно снабдить переменную определенным значением.

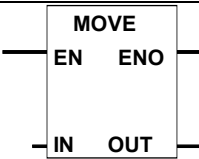
Значение, заданное на входе IN, копируется по адресу, указанному на выходе OUT. ENO имеет такое же состояние сигнала, как EN.

С помощью блока MOVE команда *Присвоить значение* может копировать все типы данных длиной 8, 16 или 32 бита. Определяемые пользователем типы данных, такие как массивы или структуры, должны копироваться при помощи встроенной системной функции Direct Word Move [прямая пересылка слова] (см. *справочное руководство I235I*).

На команду *Присвоить значение* влияет главное управляющее реле [Master Control Relay (MCR)]. Дополнительную информацию о функционировании MCR вы найдете в разделе 16.5.

На расположение блока *Присвоить значение* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–1. Блок “Присваивание значения” и параметры

Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	Все типы данных длиной 8, 16 и 32 бита	I, Q, M, D, L или константа	Исходное значение
	OUT	Все типы данных длиной 8, 16 и 32 бита	I, Q, M, D, L	Адрес назначения

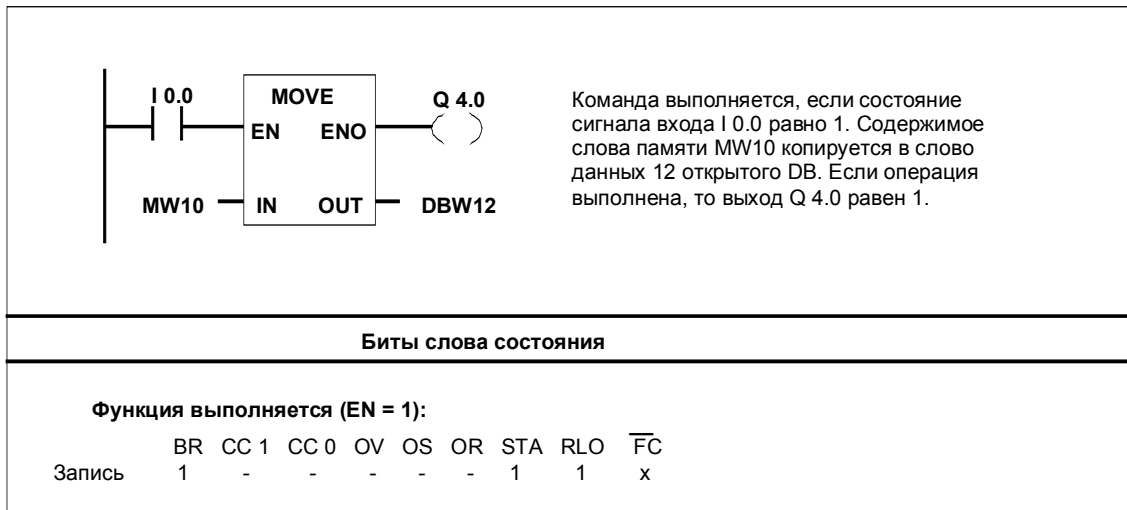


Рис. 10–1. Присваивание значения

Предварительное присваивание определенного значения переменной

Информацию о встроенных системных функциях, действующих в качестве команд пересылки, которые могут предварительно присваивать определенное значение переменной или копировать переменные различных типов, вы найдете в *Справочном руководстве I235/*.

10.2 Преобразование двоично-десятичного числа в целое

Описание

Команда *Преобразовать двоично-десятичное число в целое* считывает содержимое входного параметра IN как трехразрядное двоично-десятичное число (BCD, ± 999) и преобразует это число в целое значение. Выходной параметр OUT предоставляет результат.

ENO и EN всегда имеют одинаковое состояние сигнала.

Если разряд двоично-десятичного числа находится в недопустимом диапазоне от 10 до 15, то во время попытки преобразования возникает ошибка BCDF.

- CPU переходит в режим STOP. В диагностический буфер записывается сообщение «BCD Conversion Error [ошибка преобразования двоично-десятичного числа]» с идентификационным номером события 2521.
- Вызывается OB121, если он запрограммирован. Дополнительную информацию о программировании OB121 вы найдете в *Справочном руководстве I235I*.

На расположение блока *Преобразование двоично-десятичного числа в целое* накладываются некоторые ограничения (см. раздел 2.1).

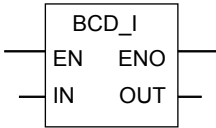
Таблица 10–2. Блок “Преобразование двоично-десятичного числа в целое” и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	WORD	I, Q, M, D, L	Двоично-десятичное число
	OUT	INT	I, Q, M, D, L	Целое значение двоично-десятичного числа



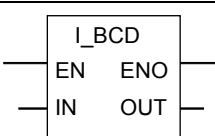
Рис. 10–2. Преобразование двоично-десятичного числа в целое

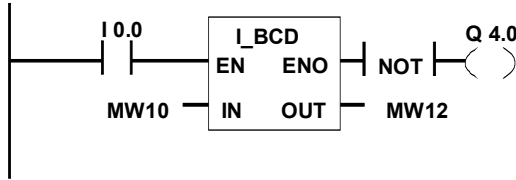
10.3 Преобразование целого числа в двоично-десятичное

Описание

Команда *Преобразовать целое число в двоично-десятичное* считывает содержимое входного параметра IN как целое значение и преобразует это значение в трехразрядное число двоично-десятичное число (BCD, ± 999). Выходной параметр OUT предоставляет результат. Если происходит переполнение, то ENO равен 0.

На расположение блока *Преобразовать целое число в двоично-десятичное* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–3. Блок “Преобразование целого числа в двоично-десятичное” и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	INT	I, Q, M, D, L	Целое число
	OUT	WORD	I, Q, M, D, L	Результат в двоично-десятичном формате



Если состояние сигнала входа I 0.0 равно 1, то преобразование выполняется. Содержимое слова памяти MW10 считывается как целое число и преобразуется в трехразрядное двоично-десятичное число. Результат сохраняется в слове памяти MW12. Если произошло переполнение, то состояние сигнала Q 4.0 равно 1. Если состояние сигнала на входе EN равно 0 (то есть преобразование не выполняется), то состояние сигнала выхода Q 4.0 также равно 1.

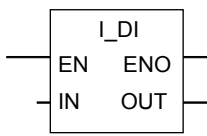
Биты слова состояния									
Функция выполняется (EN = 1):									
Запись	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
	1	-	-	x	x	0	1	x	x

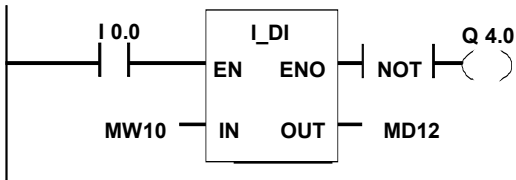
Рис. 10–3. Преобразование целого числа в двоично-десятичное

10.4 Преобразование целого числа в двойное целое

Описание

Команда *Преобразовать целое число в двойное целое* считывает содержимое входного параметра IN как целое число и преобразует это целое число в двойное целое число. Выходной параметр OUT предоставляет результат. ENO и EN всегда имеют одинаковое состояние сигнала. На расположение блока *Преобразовать целое число в двойное целое* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–4. Блок “Преобразование целого числа в двойное целое” и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	INT	I, Q, M, D, L	Преобразуемое значение
	OUT	DINT	I, Q, M, D, L	Результат



Если состояние сигнала входа I 0.0 равно 1, то преобразование выполняется. Содержимое слова памяти MW10 считывается как целое число и преобразуется в двойное целое число. Результат сохраняется в двойном слове памяти MD12. Если преобразование не выполнено, то состояние сигнала выхода Q 4.0 равно 1 (ENO = EN).

Биты слова состояния										
Функция выполняется (EN = 1):										
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC	
Запись	1	-	-	-	-	0	1	1	x	

Рис. 10–4. Преобразование целого числа в двойное целое

10.5 Преобразование двоично-десятичного числа в двойное целое

Описание

Команда *Преобразовать двоично-десятичное число в двойное целое* считывает содержимое входного параметра IN как семиразрядное двоично-десятичное число (BCD, $\pm 9\,999\,999$) и преобразует это число в двойное целое число. Выходной параметр OUT предоставляет результат.

ENO и EN всегда имеют одинаковое состояние сигнала.

Если разряд двоично-десятичного числа находится в недопустимом диапазоне от 10 до 15, то во время попытки преобразования возникает ошибка BCDF.

- CPU переходит в режим STOP. В диагностический буфер записывается сообщение «BCD Conversion Error [ошибка преобразования двоично-десятичного числа]» с идентификационным номером события 2521.
- Вызывается OB121, если он запрограммирован. Дополнительную информацию о программировании OB121 вы найдете в Справочном руководстве *I235I*.

На расположение блока *Преобразовать двоично-десятичное число в двойное целое* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–5. Блок “Преобразование двоично-десятичного числа в двойное целое” и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	DWORD	I, Q, M, D, L	Двоично-десятичное число
	OUT	DINT	I, Q, M, D, L	Значение двоично-десятичного числа в формате двойного целого числа

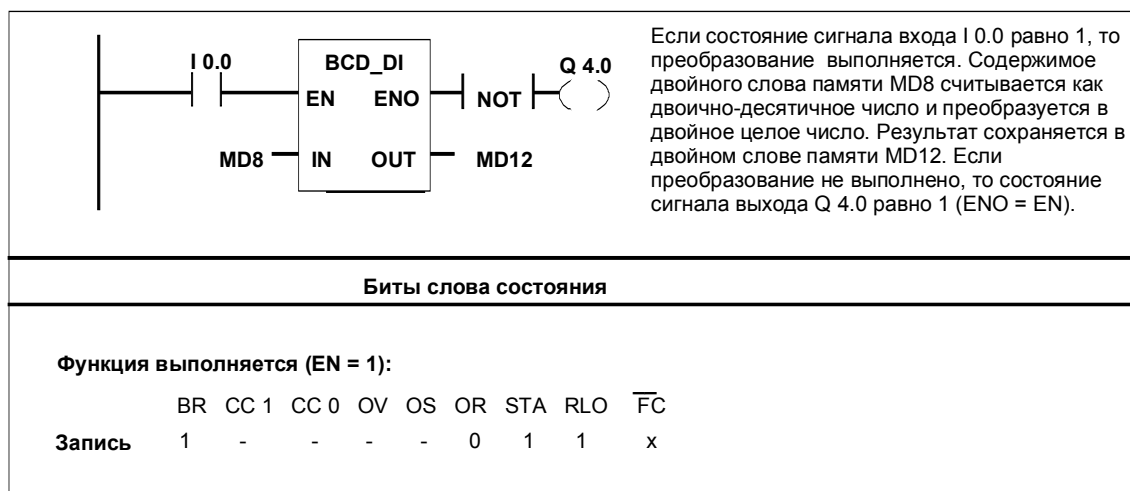


Рис. 10–5 Преобразование двоично-десятичного числа в двойное целое

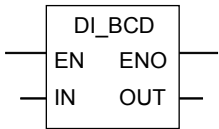
10.6 Преобразование двойного целого числа в двоично-десятичное

Описание

Команда *Преобразовать двойное целое число в двоично-десятичное* считывает содержимое входного параметра IN как двойное целое число и преобразует это значение в семиразрядное двоично-десятичное число ($\pm 9\,999\,999$). Выходной параметр OUT предоставляет результат. Если происходит переполнение, то ENO равен 0.

На расположение блока *Преобразовать двойное целое число в двоично-десятичное* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–6. Блок “Преобразование двойного целого числа в двоично-десятичное” и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	DINT	I, Q, M, D, L	Двойное целое число
	OUT	DWORD	I, Q, M, D, L	Результат в формате двоично-десятичного числа

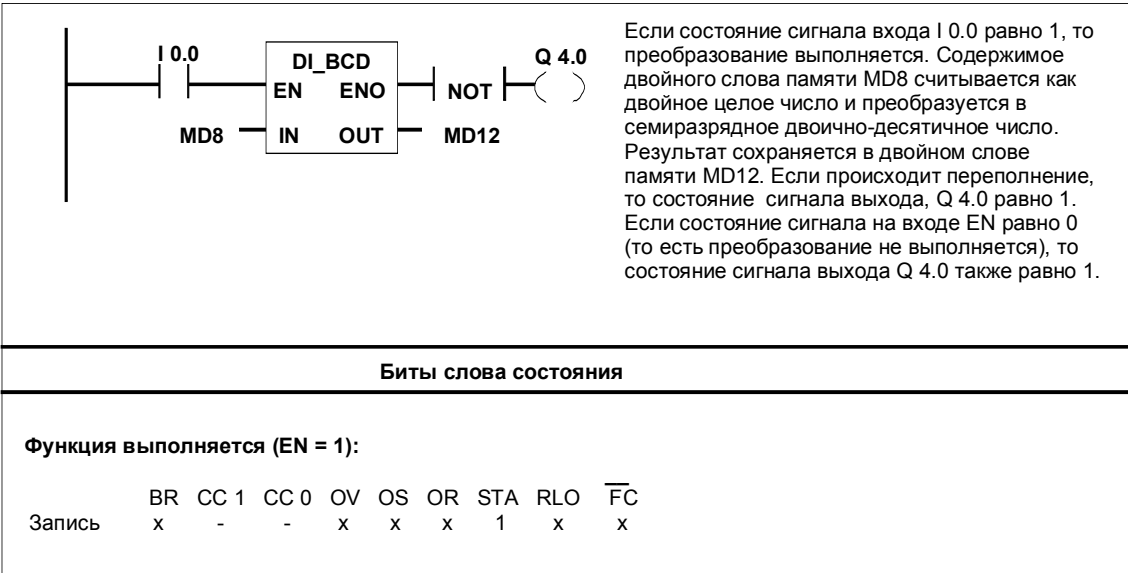


Рис. 10–6. Преобразование двойного целого числа в двоично-десятичное

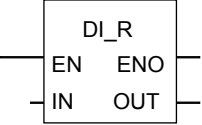
10.7 Преобразование двойного целого числа в число с плавающей точкой

Описание

Команда *Преобразовать двойное целое число в число с плавающей точкой* считывает содержимое входного параметра IN как двойное целое число и преобразует это значение в вещественное число. Выходной параметр OUT предоставляет результат. ENO и EN всегда имеют одинаковое состояние сигнала.

На расположение блока *Преобразовать двойное целое число в вещественное* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–7. Блок “Преобразование двойного целого числа в число с плавающей точкой” и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	DINT	I, Q, M, D, L	Преобразуемое значение
	OUT	REAL	I, Q, M, D, L	Результат

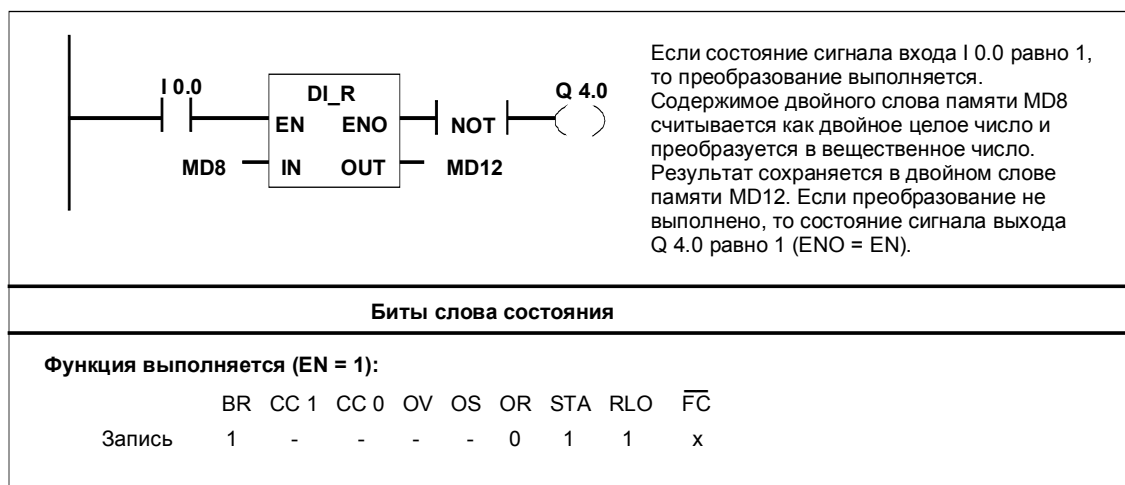


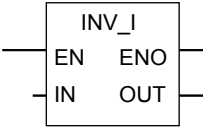
Рис. 10–7. Преобразование двойного целого числа в число с плавающей точкой

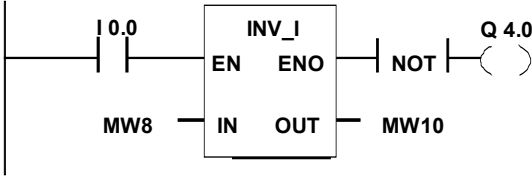
10.8 Дополнение целого числа до единицы

Описание

Команда *Дополнить целое число до единицы* считывает содержимое входного параметра IN и выполняет над этим словом и маской FFFF_H поразрядную команду булевой логики *Поразрядное исключающее ИЛИ над словами* (см. раздел 11.6), так что каждый бит слова изменяет свое значение на противоположное. Выходной параметр OUT предоставляет результат. ENO и EN всегда имеют одинаковое состояние сигнала.

На расположение блока *Дополнить целое число до единицы* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–8. Блок “Дополнение целого числа до единицы” и параметры				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	INT	I, Q, M, D, L	Входное значение
	OUT	INT	I, Q, M, D, L	Дополнение целого числа до единицы



Если состояние сигнала входа I 0.0 равно 1, то преобразование, выполняется. Каждый бит в MW8 инвертируется.
MW8 = 00000000 00000000 →
MW10 = 11111111 11111111
Если преобразование не выполнено, то состояние сигнала выхода Q 4.0 равно 1 (ENO = EN).

Биты слова состояния

Функция выполняется (EN = 1):	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	-	-	-	-	x	1	x	x

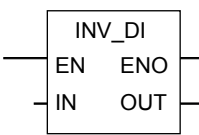
Рис. 10–8. Дополнение целого числа до единицы

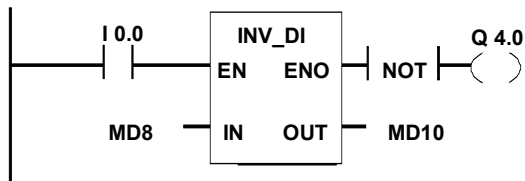
10.9 Дополнение двойного целого числа до единицы

Описание

Команда *Дополнить двойное целое число до единицы* считывает содержимое входного параметра IN и выполняет над этим словом и маской FFFF FFFF_h поразрядную команду булевой логики *Поразрядное исключающее ИЛИ над двойными словами* (см. раздел 11.6), так что каждый бит слова изменяет свое значение на противоположное. Выходной параметр OUT предоставляет результат. ENO и EN всегда имеют одинаковое состояние сигнала.

На расположение блока *Дополнить двойное целое число до единицы* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–9. Блок “Дополнение двойного целого числа до единицы” и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	DINT	I, Q, M, D, L	Входное значение
	OUT	DINT	I, Q, M, D, L	Дополнение двойного целого числа до единицы



Если состояние сигнала входа I 0.0 равно 1, то преобразование выполняется. Каждый бит двойного слова MD8 изменяется:
MD8 =FFFF FFFF → MD10 = 0000 0000.
Если преобразование не выполнено, то состояние сигнала выхода Q 4.0 равно 1 (ENO = EN).

Биты слова состояния									
Функция выполняется (EN = 1):									
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	\overline{FC}
Запись	x	-	-	-	-	x	1	x	x

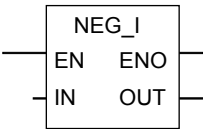
Рис. 10–9. Дополнение двойного целого числа до единицы

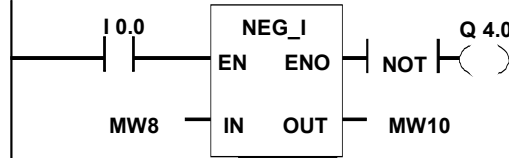
10.10 Дополнение целого числа до двух

Описание

Команда *Дополнить целое число до двух* считывает содержимое входного параметра IN и изменяет его знак (например, с положительного значения на отрицательное). Выходной параметр OUT предоставляет результат. Если состояние сигнала EN равно 0, то и состояние сигнала ENO равно 0. Если состояние сигнала EN равно 1 и происходит переполнение, то состояние сигнала ENO равно 0.

На расположение блока *Дополнить целое число до двух* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–10. Блок “Дополнение целого числа до двух” и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	INT	I, Q, M, D, L	Входное значение
	OUT	INT	I, Q, M, D, L	Дополнение целого числа до двух



Если состояние сигнала входа I 0.0 равно 1, то преобразование выполняется. Значение слова памяти MW8 через OUT передается с противоположным знаком в слово памяти MW10, как показано в следующем примере: MW8 = +10 → MW10 = - 10.

Если состояние сигнала EN равно 1 и происходит переполнение, то состояние сигнала ENO равно 0 и состояние сигнала выхода Q 4.0 равно 1. Если преобразование не выполнено, то состояние сигнала выхода Q 4.0 равно 1 (ENO = EN).

Биты слова состояния									
Функция выполняется (EN = 1):									
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	x	x	x	x	x	1	x	x

Рис. 10–10. Дополнение целого числа до двух

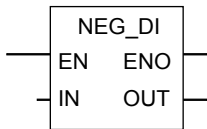
10.11 Дополнение двойного целого числа до двух

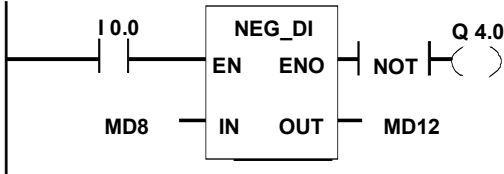
Описание

Команда *Дополнить двойное целое число до двух* считывает содержимое входного параметра IN и изменяет его знак (например, с положительного значения на отрицательное). Выходной параметр OUT предоставляет результат. Если состояние сигнала EN равно 0, то и состояние сигнала ENO равно 0. Если состояние сигнала EN равно 1 и происходит переполнение, то состояние сигнала ENO равно 0.

На расположение блока *Дополнить двойное целое число до двух* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–11. Блок “Дополнение двойного целого числа до двух” и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	DINT	I, Q, M, D, L	Входное значение
	OUT	DINT	I, Q, M, D, L	Дополнение двойного целого числа до двух



Если состояние сигнала входа I 0.0 равно 1, то преобразование выполняется. Значение двойного слова памяти MD8 через OUT передается с противоположным знаком в слово памяти MD12, как показано в следующем примере:
 MD8 = +60 000 → MD12 = - 60 000.

Если состояние сигнала EN равно 1 и происходит переполнение, то состояние сигнала ENO равно 0 и состояние сигнала выхода Q 4.0 равно 1. Если преобразование не выполнено, то состояние сигнала выхода Q 4.0 равно 1 (ENO = EN).

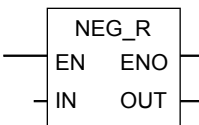
Биты слова состояния									
Функция выполняется (EN = 1):									
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	x	x	x	x	x	1	x	x

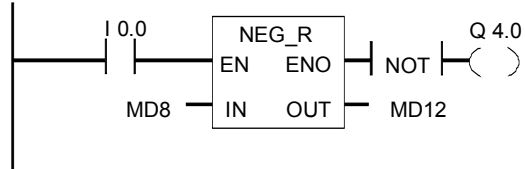
Рис. 10–11. Дополнение двойного целого числа до двух

10.12 Изменение знака числа с плавающей точкой

Команда *Изменить знак числа с плавающей точкой* считывает содержимое входного параметра IN и изменяет его знаковый бит, то есть эта команда изменяет знак числа (например, с 0 для «плюс» на 1 для «минус»). Биты порядка и мантииссы остаются неизменными. Выходной параметр OUT предоставляет результат. ENO и EN всегда имеют одинаковое состояние сигнала.

На расположение блока *Изменить знак числа с плавающей точкой* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–12. Блок “Изменение знака числа с плавающей точкой” и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Входное значение
	OUT	REAL	I, Q, M, D, L	Результат, представляющий собой входное значение с обратным знаком.



Если состояние сигнала входа I 0.0 равно 1, то преобразование выполняется. Значение двойного слова памяти MD8 через OUT передается с противоположным знаком в двойное слово памяти MD12, как показано в следующем примере:
MD8 = $+6.234 \times 10^{-3} \rightarrow$ MD12 = -6.234×10^{-3}
Если преобразование не выполнено, то состояние сигнала выхода Q 4.0 равно 1 (ENO = EN).

Биты слова состояния

Функция выполняется (EN = 1):

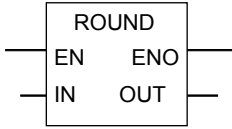
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	\overline{FC}
Запись	x	-	-	-	-	0	x	x	1

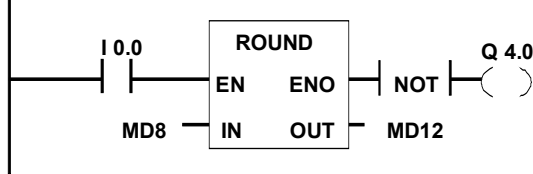
Рис. 10–12. Изменение знака числа с плавающей точкой

10.13 Округление до двойного целого числа

Описание

Команда *Округлить до двойного целого числа* считывает содержимое входного параметра IN как вещественное число и преобразует это число в двойное целое число, округляя его до ближайшего целого числа. Результатом является ближайшая целая составляющая вещественного числа (то есть ближайшее целое число). Выходной параметр OUT предоставляет результат. Если происходит переполнение, то ENO равен 0. На расположение блока *Округлить до двойного целого числа* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–13. Блок “Округление до двойного целого числа” и параметры				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Округляемое значение
	OUT	DINT	I, Q, M, D, L	Значение IN, округленное до ближайшего целого числа



Если состояние сигнала входа I 0.0 равно 1, то преобразование выполняется. Содержимое двойного слова памяти MD8 считывается как вещественное число и преобразуется в двойное целое число. Результат этой функции округления до ближайшего двойного целого числа сохраняется в двойном слове памяти MD12. Если происходит переполнение, то состояние сигнала выхода Q 4.0 равно 1. Если состояние сигнала на входе EN равно 0 (то есть преобразование не выполняется), то состояние сигнала выхода Q 4.0 также равно 1.

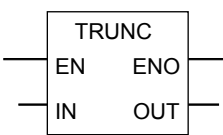
Биты слова состояния									
Функция выполняется (EN = 1):									
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	-	-	x	x	x	1	x	x

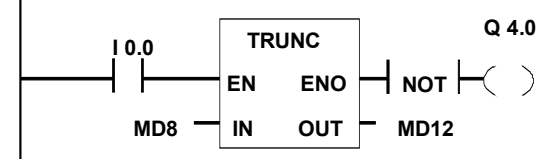
Рис. 10–13. Округление до двойного целого числа

10.14 Выделение целой части числа

Описание

Команда *Выделить целую часть числа* считывает содержимое входного параметра IN как вещественное число и преобразует это число в двойное целое число, округляя его до ближайшего меньшего или равного ему целого числа. Результатом является целая составляющая заданного вещественного числа (то есть целая часть вещественного числа). Выходной параметр OUT предоставляет результат. Если происходит переполнение, то ENO равен 0. На расположение блока *Выделить целую часть числа* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–14. Блок “Выделение целой части числа” и параметры				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Округляемое значение
	OUT	DINT	I, Q, M, D, L	Целая часть значения IN



Если состояние сигнала входа I 0.0 равно 1, то преобразование выполняется. Содержимое двойного слова памяти MD8 считывается как вещественное число и преобразуется в двойное целое число. Целая часть является результатом и сохраняется в двойном слове памяти MD12. Если происходит переполнение, то состояние сигнала выхода Q 4.0 равно 1. Если состояние сигнала на входе EN равно 0 (то есть преобразование не выполняется), то состояние сигнала выхода Q 4.0 также равно 1

Биты слова состояния									
Функция выполняется (EN = 1):									
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	-	-	x	x	x	1	x	x

Рис. 10–14. Выделение целой части числа

10.15 Округление до ближайшего большего целого числа

Описание

Команда *Округлить до ближайшего большего целого числа* считывает содержимое входного параметра IN как вещественное число и преобразует это число в двойное целое число. Результатом является наименьшее целое число, которое больше заданного вещественного числа или равно ему. Выходной параметр OUT предоставляет результат. Если происходит переполнение, то ENO равен 0.

На расположение блока *Округлить до ближайшего большего целого числа* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–15. Блок “Округление до ближайшего большего целого числа” и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Преобразуемое значение
	OUT	DINT	I, Q, M, D, L	Результат

Если состояние сигнала входа I 0.0 равно 1, то преобразование выполняется. Содержимое двойного слова памяти MD8 считывается как вещественное число и преобразуется в двойное целое число с округлением до ближайшего большего (или равного) целого числа. Результат сохраняется в двойном слове памяти MD12. Если происходит переполнение, то состояние сигнала выхода Q 4.0 равно 1. Если состояние сигнала на входе EN равно 0 (то есть преобразование не выполняется), то состояние сигнала выхода Q 4.0 также равно 1.

Биты слова состояния									
Функция выполняется (EN = 1):									
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	-	-	x	x	x	1	x	x

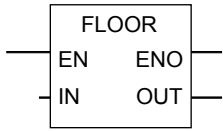
Рис. 10–15. Округление до ближайшего большего целого числа

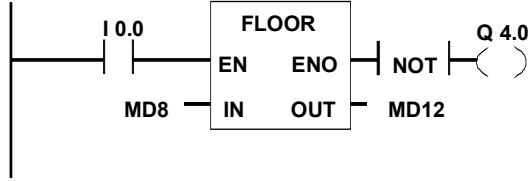
10.16 Округление до ближайшего меньшего целого числа

Описание

Команда *Округлить до ближайшего меньшего целого числа* считывает содержимое входного параметра IN как вещественное число и преобразует это число в двойное целое число. Результатом является наибольшее целое число, которое меньше заданного вещественного числа или равно ему. Выходной параметр OUT предоставляет результат. Если происходит переполнение, то ENO равен 0.

На расположение блока *Округлить до ближайшего меньшего целого числа* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 10–16. Блок “Округление до ближайшего меньшего целого числа” и параметры				
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	REAL	I, Q, M, D, L	Преобразуемое значение
	OUT	DINT	I, Q, M, D, L	Результат



Если состояние сигнала входа I 0.0 равно 1, то преобразование выполняется. Содержимое двойного слова памяти MD8 считывается как вещественное число и преобразуется в двойное целое число с округлением до ближайшего меньшего (или равного) целого числа. Результат сохраняется в двойном слове памяти MD12. Если происходит переполнение, то состояние сигнала выхода Q 4.0 равно 1. Если состояние сигнала на входе EN равно 0 (то есть преобразование не выполняется), то состояние сигнала выхода Q 4.0 также равно 1.

Биты слова состояния

Функция выполняется (EN = 1):

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	x	-	-	x	x	x	1	x	x

Рис. 10–16. Округление до ближайшего меньшего целого числа

11 Поразрядные логические операции над словами

Обзор главы

Раздел	Описание	Страница
11.1	Обзор	11–2
11.2	Поразрядное И над словами	11–3
11.3	Поразрядное И над двойными словами	11–5
11.4	Поразрядное ИЛИ над словами	11–7
11.5	Поразрядное ИЛИ над двойными словами	11–9
11.6	Поразрядное исключающее ИЛИ над словами	11–11
11.7	Поразрядное исключающее ИЛИ над двойными словами	11–13

11.1 Обзор

Что такое поразрядные логические команды над словами?

Поразрядные логические команды над словами побитно сравнивают пары слов (16 бит) или двойных слов (32 бита) в соответствии с булевой логикой. Для выполнения поразрядных логических операций над словами имеются в распоряжении следующие команды:

- Поразрядное И над словами: побитно объединяет два слова в соответствии с таблицей истинности логической операции И.
- Поразрядное И над двойными словами: побитно объединяет два двойных слова в соответствии с таблицей истинности логической операции И.
- Поразрядное ИЛИ над словами: побитно объединяет два слова в соответствии с таблицей истинности логической операции ИЛИ.
- Поразрядное ИЛИ над двойными словами: побитно объединяет два двойных слова в соответствии с таблицей истинности логической операции ИЛИ.
- Поразрядное исключающее ИЛИ над словами: побитно объединяет два слова в соответствии с таблицей истинности логической операции ИСКЛЮЧАЮЩЕЕ ИЛИ.
- Поразрядное исключающее ИЛИ над двойными словами: побитно объединяет два двойных слова в соответствии с таблицей истинности логической операции ИСКЛЮЧАЮЩЕЕ ИЛИ.

11.2 Поразрядное И над словами

Описание

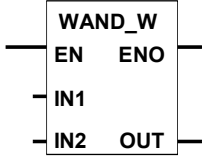
Сигнал 1 на разрешающем входе (EN) активизирует команду *Поразрядное И над словами*. Эта команда побитно объединяет два цифровых значения, указанные на входах IN1 и IN2, в соответствии с таблицей истинности логической операции И. Эти значения интерпретируются как чисто битовые комбинации. Результат можно просмотреть на выходе OUT. ENO имеет такое же состояние сигнала, как EN.

Отношение результата на выходе OUT к 0 влияет на бит кода условия CC 1 слова состояния следующим образом:

- Если результат на выходе OUT не равен 0, то бит кода условия CC 1 слова состояния устанавливается в 1.
- Если результат на выходе OUT равен 0, то бит кода условия CC 1 слова состояния равен 0.

На расположение блоков поразрядных логических операций над словами накладываются некоторые ограничения (см. раздел 2.1).

Таблица 11–1. Блок “Поразрядное И над словами” и параметры

Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	WORD	I, Q, M, D, L	Первый операнд логической операции
	IN2	WORD	I, Q, M, D, L	Второй операнд логической операции
	OUT	WORD	I, Q, M, D, L	Результат логической операции

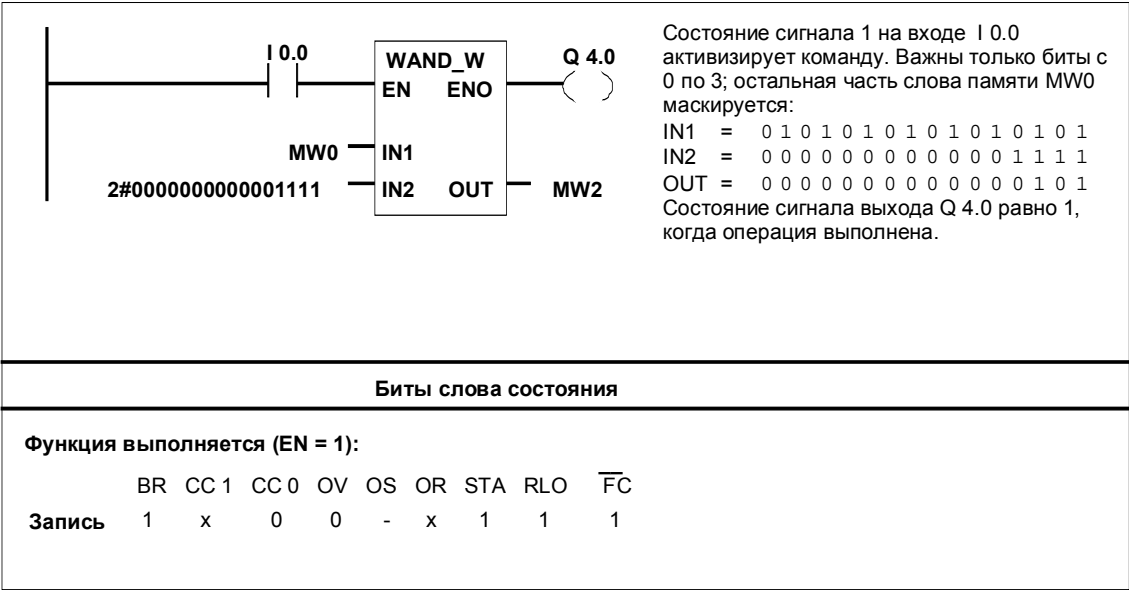


Рис. 11–1. Поразрядное И над словами

11.3 Поразрядное И над двойными словами

Описание

Сигнал 1 на разрешающем входе (EN) активизирует команду *Поразрядное И над двойными словами*. Эта команда побитно объединяет два цифровых значения, указанные на входах IN1 и IN2, в соответствии с таблицей истинности логической операции И. Эти значения интерпретируются как чисто битовые комбинации. Результат можно просмотреть на выходе OUT. ENO имеет такое же состояние сигнала, как EN.

Отношение результата на выходе OUT к 0 влияет на бит кода условия CC 1 слова состояния следующим образом:

- Если результат на выходе OUT не равен 0, то бит кода условия CC 1 слова состояния устанавливается в 1.
- Если результат на выходе OUT равен 0, то бит кода условия CC 1 слова состояния равен 0.

На расположение блоков поразрядных логических операций над словами накладываются некоторые ограничения (см. раздел 2.1).

Таблица 11–2. Блок “Поразрядное И над двойными словами” и параметры

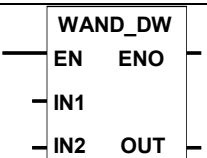
Блок KOP	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	DWORD	I, Q, M, D, L	Первый операнд логической операции
	IN2	DWORD	I, Q, M, D, L	Второй операнд логической операции
	OUT	DWORD	I, Q, M, D, L	Результат логической операции



Рис. 11–2. Поразрядное И над двойными словами

11.4 Поразрядное ИЛИ над словами

Описание

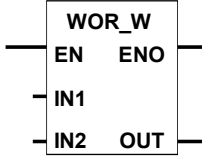
Сигнал 1 на разрешающем входе (EN) активизирует команду *Поразрядное ИЛИ над словами*. Эта команда побитно объединяет два цифровых значения, указанные на входах IN1 и IN2, в соответствии с таблицей истинности логической операции ИЛИ. Эти значения интерпретируются как чисто битовые комбинации. Результат можно просмотреть на выходе OUT. ENO имеет такое же состояние сигнала, как EN.

Отношение результата на выходе OUT к 0 влияет на бит кода условия CC 1 слова состояния следующим образом:

- Если результат на выходе OUT не равен 0, то бит кода условия CC 1 слова состояния устанавливается в 1.
- Если результат на выходе OUT равен 0, то бит кода условия CC 1 слова состояния равен 0.

На расположение блоков поразрядных логических операций над словами накладываются некоторые ограничения (см. раздел 2.1).

Таблица 11–3. Блок “Поразрядное ИЛИ над словами” и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	WORD	I, Q, M, D, L	Первый операнд логической операции
	IN2	WORD	I, Q, M, D, L	Второй операнд логической операции
	OUT	WORD	I, Q, M, D, L	Результат логической операции

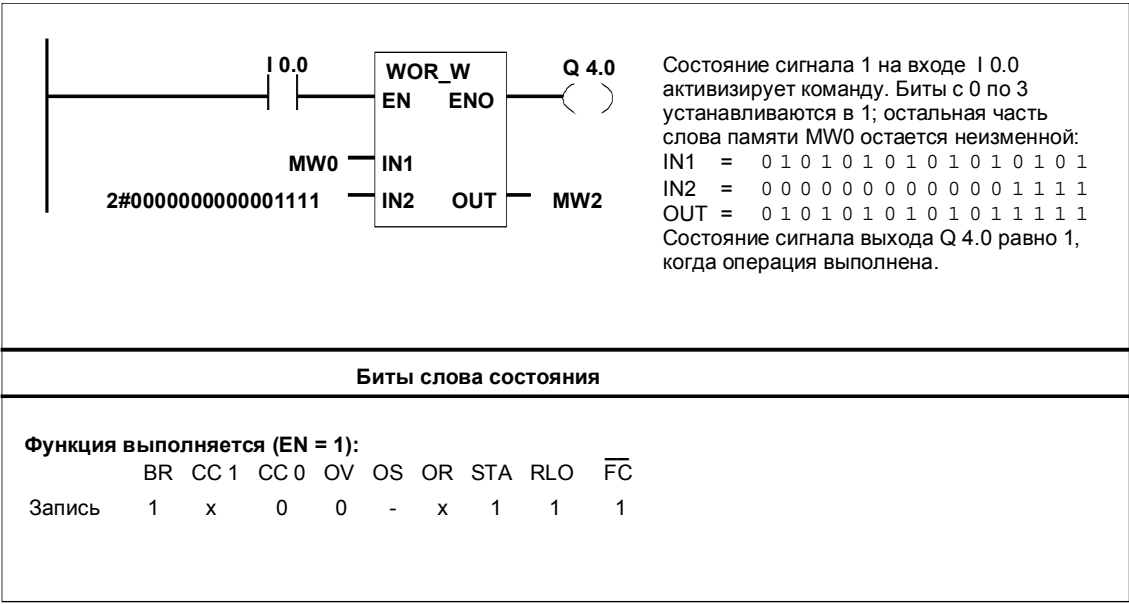


Рис. 11–3 Поразрядное ИЛИ над словами

11.5 Поразрядное ИЛИ над двойными словами

Описание

Сигнал 1 на разрешающем входе (EN) активизирует команду *Поразрядное ИЛИ над двойными словами*. Эта команда побитно объединяет два цифровых значения, указанные на входах IN1 и IN2, в соответствии с таблицей истинности логической операции ИЛИ. Эти значения интерпретируются как чисто битовые комбинации. Результат можно просмотреть на выходе OUT. ENO имеет такое же состояние сигнала, как EN.

Отношение результата на выходе OUT к 0 влияет на бит кода условия CC 1 слова состояния следующим образом:

- Если результат на выходе OUT не равен 0, то бит кода условия CC 1 слова состояния устанавливается в 1.
- Если результат на выходе OUT равен 0, то бит кода условия CC 1 слова состояния равен 0.

На расположение блоков поразрядных логических операций над словами накладываются некоторые ограничения (см. раздел 2.1).

Таблица 11–4. Блок “Поразрядное ИЛИ над двойными словами” и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN1	DWORD	I, Q, M, D, L	Первый операнд логической операции
	IN2	DWORD	I, Q, M, D, L	Второй операнд логической операции
	OUT	DWORD	I, Q, M, D, L	Результат логической операции

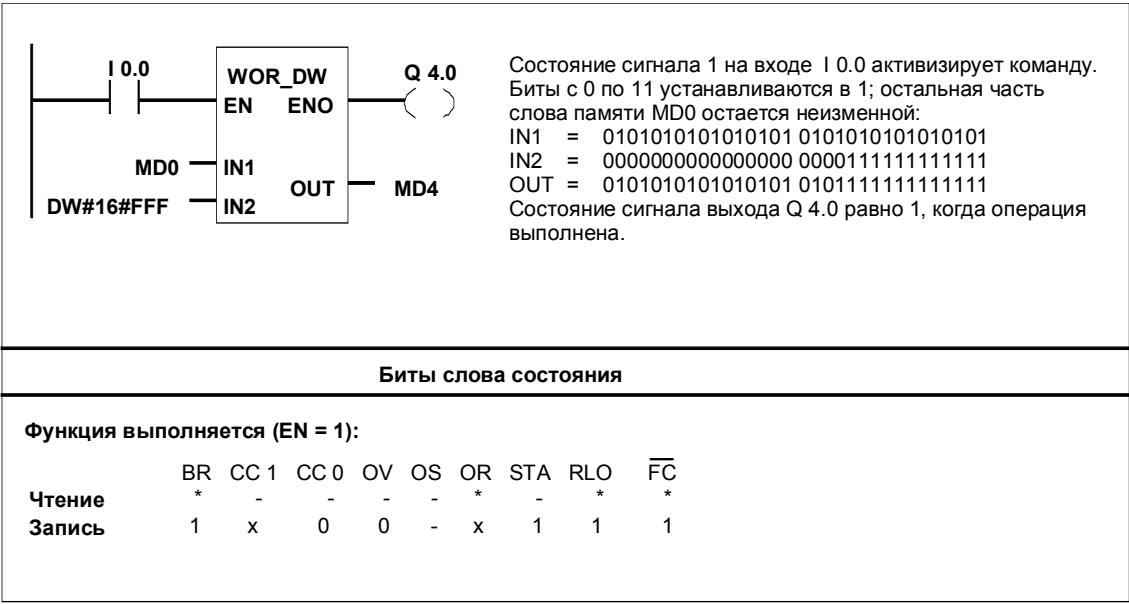


Рис. 11–4. Поразрядное ИЛИ над двойными словами

11.6 Поразрядное исключающее ИЛИ над словами

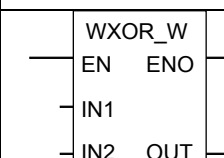
Описание

Сигнал 1 на разрешающем входе (EN) активизирует команду *Поразрядное исключающее ИЛИ над словами*. Эта команда побитно объединяет два цифровых значения, указанные на входах IN1 и IN2, в соответствии с таблицей истинности логической операции ИСКЛЮЧАЮЩЕЕ ИЛИ. Эти значения интерпретируются как чисто битовые комбинации. Результат можно просмотреть на выходе OUT. ENO имеет такое же состояние сигнала, как EN. Отношение результата на выходе OUT к 0 влияет на бит кода условия CC 1 слова состояния следующим образом:

- Если результат на выходе OUT не равен 0, то бит кода условия CC 1 слова состояния устанавливается в 1.
- Если результат на выходе OUT равен 0, то бит кода условия CC 1 слова состояния равен 0.

На расположение блоков поразрядных логических операций над словами накладываются некоторые ограничения (см. раздел 2.1).

Таблица 11–5. Блок “Поразрядное исключающее ИЛИ над словами” и параметры

Блок КОР		Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход	
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход	
	IN1	WORD	I, Q, M, D, L	Первый операнд логической операции	
	IN2	WORD	I, Q, M, D, L	Второй операнд логической операции	
	O	WORD	I, Q, M, D, L	Результат логической операции	



Состояние сигнала 1 на входе I 0.0 активизирует команду.

IN1 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

IN2 = 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1

OUT = 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0

Состояние сигнала выхода Q 4.0 равно 1, когда операция выполнена.

Биты слова состояния

Функция выполняется (EN = 1):

BR

CC 1

CC 0

OV

OS

OR

STA

RLO

FC

Запись

1

x

0

0

-

x

1

1

1

Рис. 11–5. Поразрядное исключающее ИЛИ над словами

11.7 Поразрядное исключающее ИЛИ над двойными словами

Описание

Сигнал 1 на разрешающем входе (EN) активизирует команду *Поразрядное исключающее ИЛИ над двойными словами*. Эта команда побитно объединяет два цифровых значения, указанные на входах IN1 и IN2, в соответствии с таблицей истинности логической операции ИСКЛЮЧАЮЩЕЕ ИЛИ. Эти значения интерпретируются как чисто битовые комбинации. Результат можно просмотреть на выходе OUT. ENO имеет такое же состояние сигнала, как EN. Отношение результата на выходе OUT к 0 влияет на бит кода условия CC 1 слова состояния следующим образом:

- Если результат на выходе OUT не равен 0, то бит кода условия CC 1 слова состояния устанавливается в 1.
- Если результат на выходе OUT равен 0, то бит кода условия CC 1 слова состояния равен 0.

На расположение блоков поразрядных логических операций над словами накладываются некоторые ограничения (см. раздел 2.1).

Таблица 11–6. Блок “Поразрядное исключающее ИЛИ над двойными словами” и параметры

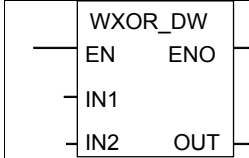
Блок КОР		Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход	
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход	
	IN1	DWORD	I, Q, M, D, L	Первый операнд логической операции	
	IN2	DWORD	I, Q, M, D, L	Второй операнд логической операции	
	OUT	DWORD	I, Q, M, D, L	Результат логической операции	



Рис. 11–6. Поразрядное исключающее ИЛИ над двойными словами

12 Команды сдвига и циклического сдвига

Обзор главы

Раздел	Описание	Страница
12.1	Команды сдвига	12–2
12.2	Команды циклического сдвига	12–11

12.1 Команды сдвига

Команды сдвига

Вы можете использовать команды сдвига, чтобы сдвигать содержимое входа IN бит за битом влево или вправо. Сдвиг влево умножает содержимое входа IN на 2 в степени n (2^n); сдвиг вправо делит содержимое входа IN на 2 в степени n (2^n). Например, если вы сдвигаете двоичный эквивалент десятичного числа 3 влево на 3 бита, то получаете в аккумуляторе двоичный эквивалент десятичного числа 24. Если вы сдвигаете двоичный эквивалент десятичного числа 16 вправо на 2 бита, то получаете в аккумуляторе двоичный эквивалент десятичного числа 4.

Число, которым вы снабжаете входной параметр N, определяет количество битов, на которое происходит сдвиг. Двоичные разряды, освобождаемые командой сдвига, заполняются или нулями, или состоянием сигнала знакового бита (0 означает положительное значение, а 1 означает отрицательное значение). Состояние сигнала бита, сдвигаемого последним, загружается в бит CC 1 слова состояния (см. раздел 2.3). Биты CC 0 и OV слова состояния сбрасываются в 0. Для анализа бита CC 1 вы можете использовать команды перехода.

Доступны следующие команды сдвига:

- Сдвинуть влево слово, Сдвинуть влево двойное слово
- Сдвинуть вправо слово, Сдвинуть вправо двойное слово
- Сдвинуть вправо целое число, Сдвинуть вправо двойное целое число

Сдвиг влево слова

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Сдвинуть влево слово*. Эта команда поочередно сдвигает влево биты с 15 по 0 входа IN. Перенос в бит 16 не происходит.

Вход N определяет количество битов, на которое происходит сдвиг. Если N больше, чем 16, то команда записывает 0 в младшее слово аккумулятора 1 и сбрасывает биты CC 0 и OV слова состояния в 0. Двоичные разряды справа заполняются нулями. Результат операции сдвига может быть просмотрен на выходе OUT.

Операция, запускаемая этой командой, всегда сбрасывает биты CC 0 и OV слова состояния в 0. ENO имеет такое же состояние сигнала, как EN.

На расположение блока *Сдвинуть влево слово* накладываются некоторые ограничения (см. раздел 2.1).

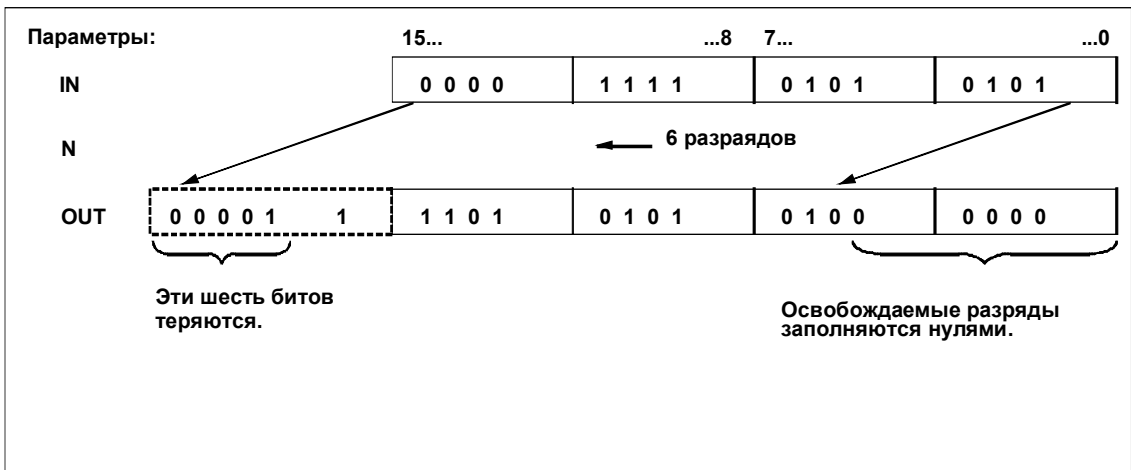


Таблица 12–1. Блок “Сдвиг влево слова” и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
<div> <div>SHL_W</div> <div>EN ENO</div> <div>IN OUT</div> <div>N</div> </div>	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	WORD	I, Q, M, D, L	Значение, подвергаемое сдвигу
	N	WORD	I, Q, M, D, L	Количество двоичных разрядов, на которое происходит сдвиг
	OUT	WORD	I, Q, M, D, L	Результат операции сдвига

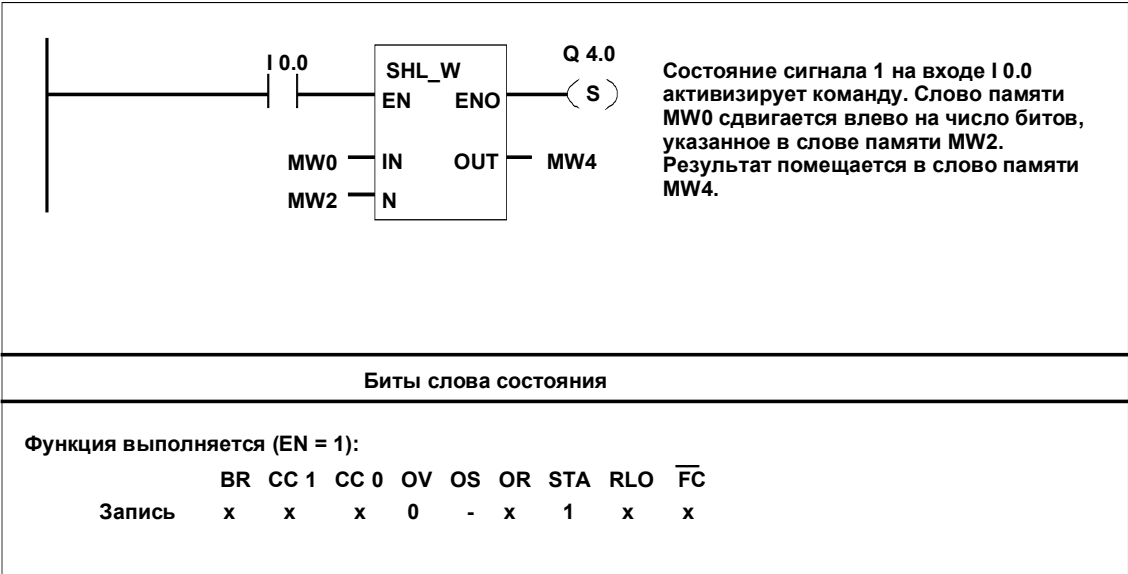


Рис. 12–1. Сдвиг влево слова

Сдвиг влево двойного слова

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Сдвинуть влево двойное слово*. Эта команда поочередно сдвигает влево биты с 31 по 0 входа IN. Вход N определяет количество битов, на которое происходит сдвиг. Если N больше, чем 32, то команда записывает 0 в выход OUT и сбрасывает биты CC 0 и OV слова состояния в 0. Двоичные разряды справа заполняются нулями. Результат операции сдвига может быть просмотрен на выходе OUT.

Операция, запускаемая этой командой, всегда сбрасывает биты CC 0 и OV слова состояния в 0, если N не равно 0. ENO имеет такое же состояние сигнала, как EN.

На расположение блока *Сдвинуть влево двойное слово* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 12–2. Блок “Сдвиг влево двойного слова” и параметры					
Блок KOP		Параметр	Тип данных	Область памяти	Описание
		EN	BOOL	I, Q, M, D, L	Разрешающий вход
		ENO	BOOL	I, Q, M, D, L	Разрешающий выход
		IN	DWORD	I, Q, M, D, L	Значение, подвергаемое сдвигу
		N	WORD	I, Q, M, D, L	Количество двоичных разрядов, на которое происходит сдвиг
		OUT	DWORD	I, Q, M, D, L	Результат операции сдвига

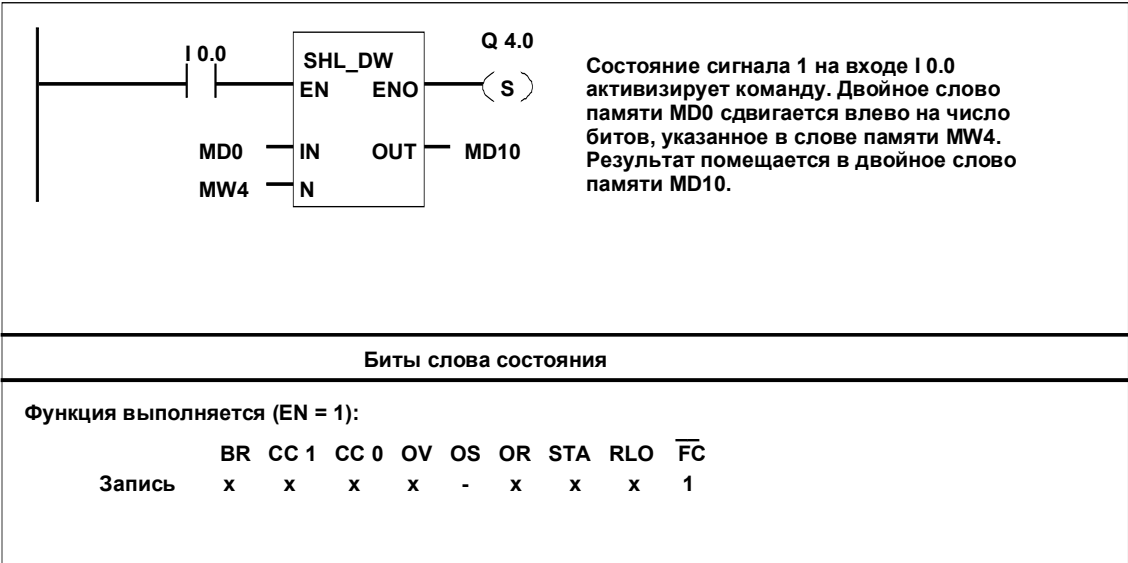


Рис. 12–2 Сдвиг влево двойного слова

Сдвиг вправо слова

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Сдвинуть вправо слово*. Эта команда поочередно сдвигает вправо биты с 0 по 15 входа IN. Биты с 16 по 31 воздействию не подвергаются. Вход N определяет количество битов, на которое происходит сдвиг. Если N больше, чем 16, то команда записывает 0 в выход OUT и сбрасывает биты CC 0 и OV слова состояния в 0. Двоичные разряды слева заполняются нулями. Результат операции сдвига может быть просмотрен на выходе OUT. Операция, запускаемая этой командой, всегда сбрасывает биты CC 0 и OV слова состояния в 0, если N не равно 0. ENO имеет такое же состояние сигнала, как EN.

На расположение блока *Сдвинуть вправо слово* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 12–3. Блок “Сдвиг вправо слова” и параметры					
Блок КОР	Параметр	Тип данных	Область памяти	Описание	
	EN	BOOL	I, Q, M, D, L	Разрешающий вход	
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход	
	IN	WORD	I, Q, M, D, L	Значение, подвергаемое сдвигу	
	N	WORD	I, Q, M, D, L	Количество двоичных разрядов, на которое происходит сдвиг	
	OUT	WORD	I, Q, M, D, L	Результат операции сдвига	

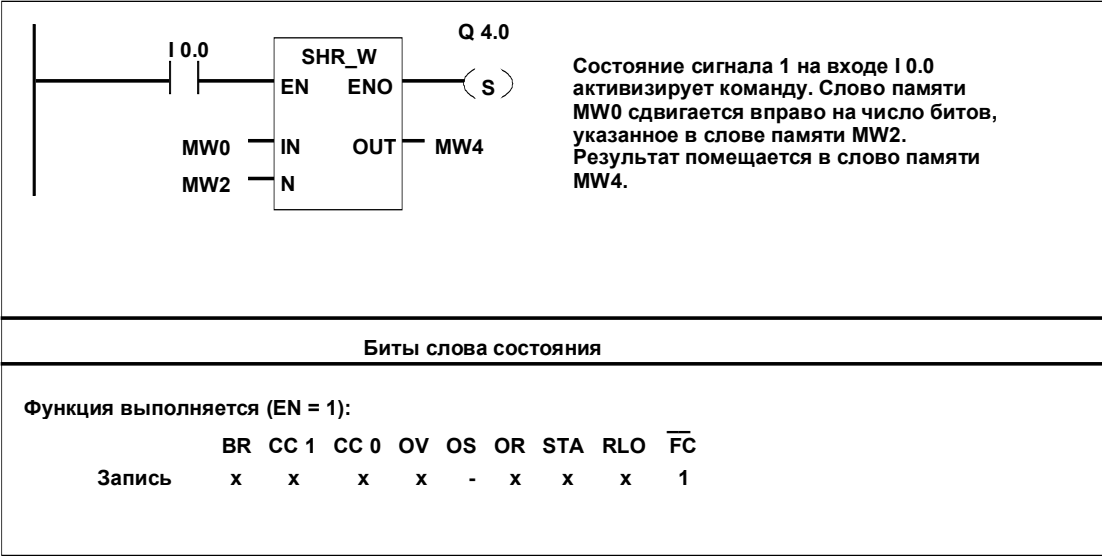


Рис. 12–3 Сдвиг вправо слова

Сдвиг вправо двойного слова

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Сдвинуть вправо двойное слово*. Эта команда поочередно сдвигает вправо биты с 0 по 31 входа IN. Вход N определяет количество битов, на которое происходит сдвиг. Если N больше, чем 32, то команда записывает 0 в выход OUT и сбрасывает биты CC 0 и OV слова состояния в 0. Двоичные разряды слева заполняются нулями. Результат операции сдвига может быть просмотрен на выходе OUT.

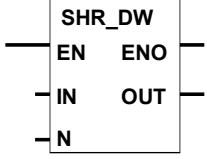
Операция, запускаемая этой командой, всегда сбрасывает биты CC 0 и OV слова состояния в 0, если N не равно 0. ENO имеет такое же состояние сигнала, как EN.

На расположение блока *Сдвинуть вправо двойное слово* накладываются некоторые ограничения (см. раздел 2.1).



Рис. 12–4. Сдвиг битов входа IN на три разряда вправо

Таблица 12–4. Блок “Сдвиг вправо двойного слова” и параметры

Блок КОР	Параметр	Тип данных	Область памяти	Описание
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	DWORD	I, Q, M, D, L	Значение, подвергаемое сдвигу
	N	WORD	I, Q, M, D, L	Количество двоичных разрядов, на которое происходит сдвиг
	OUT	DWORD	I, Q, M, D, L	Результат операции сдвига

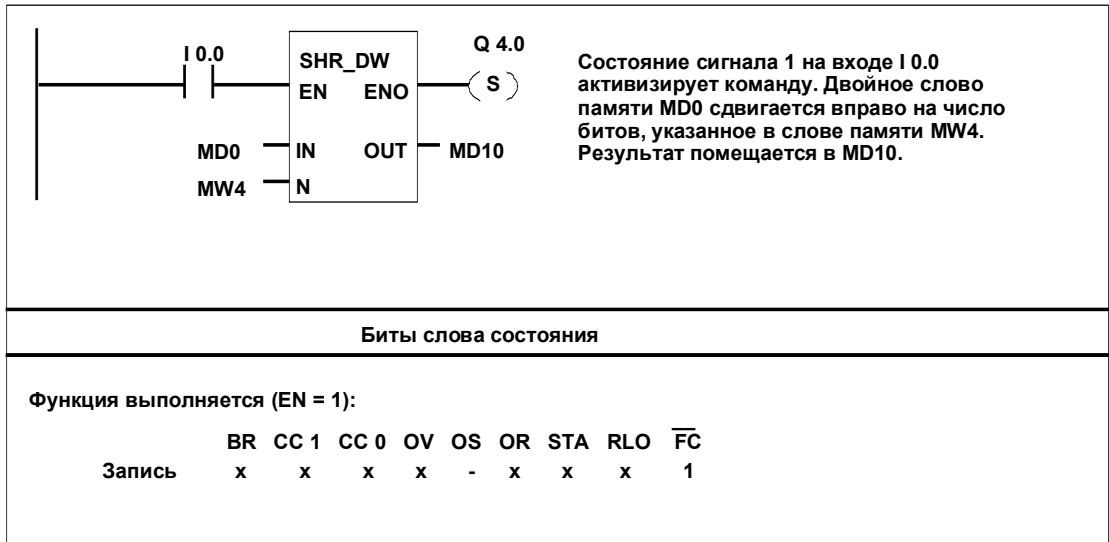


Рис. 12–5. Сдвиг вправо двойного слова

Сдвиг вправо целого числа

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Сдвинуть вправо целое число*. Эта команда поочередно сдвигает вправо биты с 0 по 15 входа IN. Вход N определяет количество битов, на которое происходит сдвиг. Если N больше, чем 16, то команда ведет себя так, как если бы N было равно 16. Двоичные разряды слева заполняются в соответствии с состоянием сигнала бита 15 (который является битом знака целого числа), то есть они заполняются нулями, если число является положительным, и единицами, если число является отрицательным. Результат операции сдвига может быть просмотрен на выходе OUT. Операция, запускаемая этой командой, всегда сбрасывает биты CC 0 и OV слова состояния в 0, если N не равно 0. ENO имеет такое же состояние сигнала, как EN. На расположение блока *Сдвинуть вправо целое число* накладываются некоторые ограничения (см. раздел 2.1).



Рис. 12–6. Сдвиг битов входа IN на четыре разряда вправо со знаком

Таблица 12–5. Блок “Сдвиг вправо целого числа” и параметры					
Блок КОР		Параметр	Тип данных	Область памяти	Описание
		EN	BOOL	I, Q, M, D, L	Разрешающий вход
		ENO	BOOL	I, Q, M, D, L	Разрешающий выход
		IN	INT	I, Q, M, D, L	Значение, подвергаемое сдвигу
		N	WORD	I, Q, M, D, L	Количество двоичных разрядов, на которое происходит сдвиг
		OUT	INT	I, Q, M, D, L	Результат операции сдвига

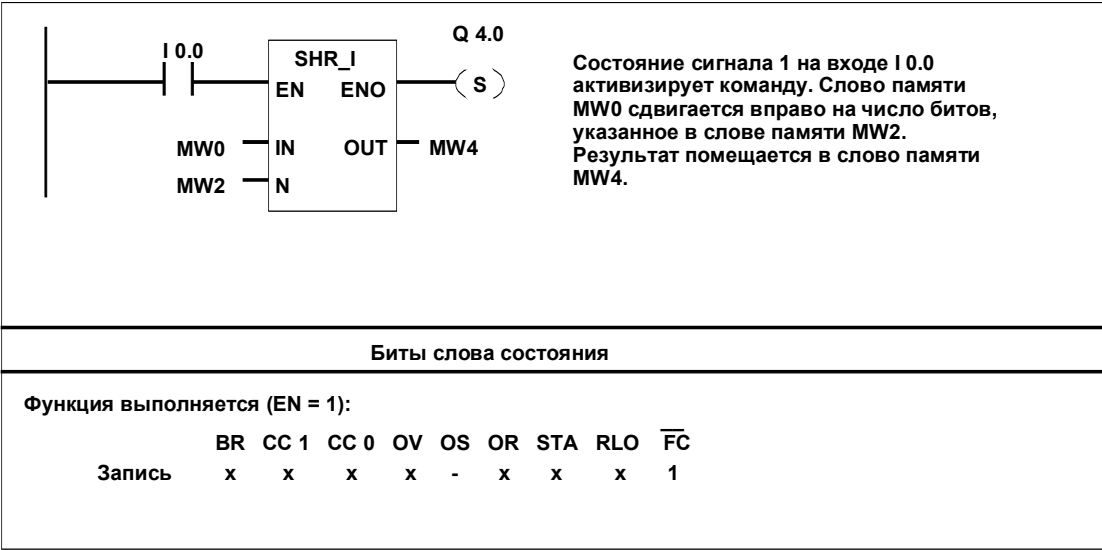


Рис. 12–7. Сдвиг вправо целого числа

Сдвиг вправо двойного целого числа

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Сдвинуть вправо двойное целое число*. Эта команда сдвигает вправо бит за битом все содержимое входа IN. Вход N определяет количество битов, на которое происходит сдвиг. Если N больше, чем 32, то команда ведет себя так, как если бы N было равно 32. Двоичные разряды слева заполняются в соответствии с состоянием сигнала бита 31 (который является битом знака двойного целого числа), то есть они заполняются нулями, если число является положительным, и единицами, если число является отрицательным. Результат операции сдвига может быть просмотрен на выходе OUT.

Операция, запускаемая этой командой, всегда сбрасывает биты CC 0 и OV слова состояния в 0, если N не равно 0. ENO имеет такое же состояние сигнала, как EN.

На расположение блока *Сдвинуть вправо двойное целое число* накладываются некоторые ограничения (см. раздел 2.1).

Таблица 12–6. Блок “Сдвиг вправо двойного целого числа” и параметры					
Блок КОР		Параметр	Тип данных	Область памяти	Описание
	SHR_DI	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	ENO	BOOL	I, Q, M, D, L	Разрешающий выход
	IN	IN	DINT	I, Q, M, D, L	Значение, подвергаемое сдвигу
	N	N	WORD	I, Q, M, D, L	Количество двоичных разрядов, на которое происходит сдвиг
	OUT	OUT	DINT	I, Q, M, D, L	Результат операции сдвига

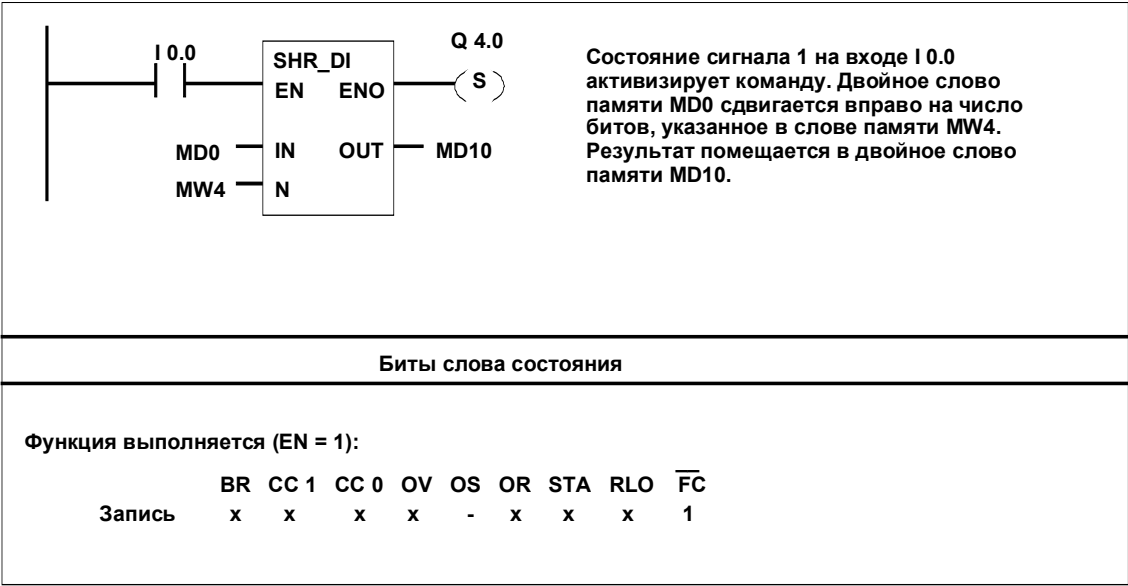


Рис. 12–8 Сдвиг вправо двойного целого числа

12.2 Команды циклического сдвига

Описание

Вы можете использовать команды циклического сдвига для того, чтобы выполнять циклический сдвиг всего содержимого входа IN бит за битом влево или вправо. Освобождаемые разряды заполняются состояниями сигналов тех битов, которые выталкиваются из содержимого входа IN.

Число, которым вы снабжаете входной параметр N, определяет число битов, на которое происходит циклический сдвиг.

В зависимости от команды циклический сдвиг происходит через бит CC 1 слова состояния (см. раздел 2.3). Бит CC 0 слова состояния сбрасывается в 0.

Имеются в распоряжении следующие команды циклического сдвига:

- Циклический сдвиг двойного слова влево
- Циклический сдвиг двойного слова вправо

Циклический сдвиг влево двойного слова

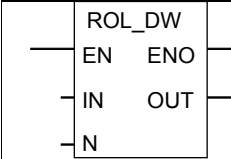
Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Выполнить циклический сдвиг двойного слова влево*. Эта команда циклически сдвигает влево бит за битом все содержимое входа IN. Вход N определяет количество битов, на которое происходит сдвиг. Если N больше, чем 32, то двойное слово циклически сдвигается на количество разрядов, равное $((N-1) \text{ по модулю } 32) + 1$. Двоичные разряды справа заполняются состояниями сигналов циклически сдвигаемых битов. Результат операции циклического сдвига может быть просмотрен на выходе OUT.

Операция, запускаемая этой командой, всегда сбрасывает биты CC 0 и OV слова состояния в 0, если N не равно 0. ENO имеет такое же состояние сигнала, как EN.

На расположение блока *Выполнить циклический сдвиг двойного слова влево* накладываются некоторые ограничения (см. раздел 2.1).



Рис. 12–9. Циклический сдвиг битов входа IN на три бита влево

Таблица 12–7. Блок “Циклический сдвиг влево двойного слова” и параметры					
Блок КОР	Параметр	Тип данных	Область памяти	Описание	
	EN	BOOL	I, Q, M, D, L	Разрешающий вход	
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход	
	IN	DWORD	I, Q, M, D, L	Значение, подвергаемое циклическому сдвигу	
	N	WORD	I, Q, M, D, L	Количество двоичных разрядов, на которое происходит циклический сдвиг	
	OUT	DWORD	I, Q, M, D, L	Результат операции циклического сдвига	

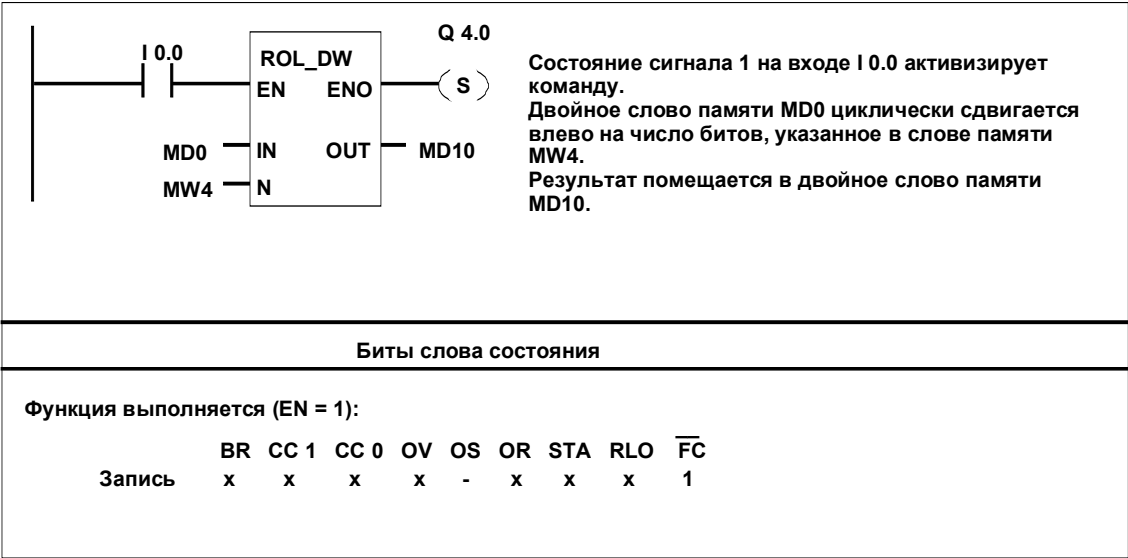


Рис. 12–10. Циклический сдвиг влево двойного слова

Циклический сдвиг вправо двойного слова

Состояние сигнала 1 на разрешающем входе (EN) активизирует команду *Выполнить циклический сдвиг двойного слова вправо*. Эта команда циклически сдвигает вправо бит за битом полное содержимое входа IN. Вход N определяет количество битов, на которое происходит сдвиг. Значение N может располагаться между 0 и 31. Если N больше, чем 32, то двойное слово циклически сдвигается на количество разрядов, равное ((N-1) по модулю 32) +1). Двоичные разряды слева заполняются состояниями сигналов циклически сдвигаемых битов. Результат операции циклического сдвига может быть просмотрен на выходе OUT.

Операция, запускаемая этой командой, всегда сбрасывает биты CC 0 и OV слова состояния в 0, если N не равно 0. ENO имеет такое же состояние сигнала, как EN.

На расположение блока *Выполнить циклический сдвиг двойного слова вправо* накладываются некоторые ограничения (см. раздел 2.1).

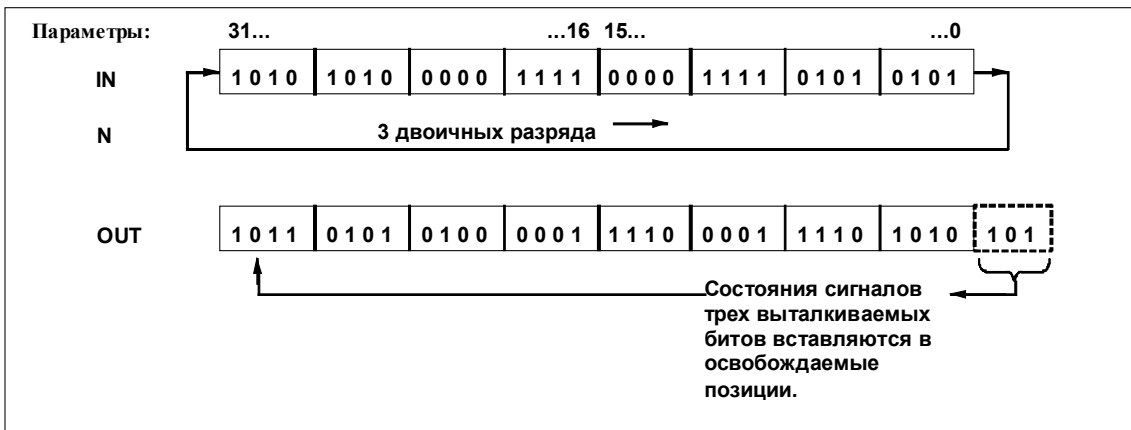


Рис. 12–11. Циклический сдвиг битов входа IN на три бита вправо

Таблица 12–8. Блок “Циклический сдвиг вправо двойного слова” и параметры					
Блок КОР	Параметр	Тип данных	Область памяти	Описание	
<div> <div>ROR_DW</div> <div> <div>EN</div> <div>ENO</div> </div> <div> <div>IN</div> <div>OUT</div> </div> <div>N</div> </div>	EN	BOOL	I, Q, M, D, L	Разрешающий вход	
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход	
	IN	DWORD	I, Q, M, D, L	Значение, подвергаемое циклическому сдвигу	
	N	WORD	I, Q, M, D, L	Количество двоичных разрядов, на которое происходит циклический сдвиг	
	OUT	DWORD	I, Q, M, D, L	Результат операции циклического сдвига	

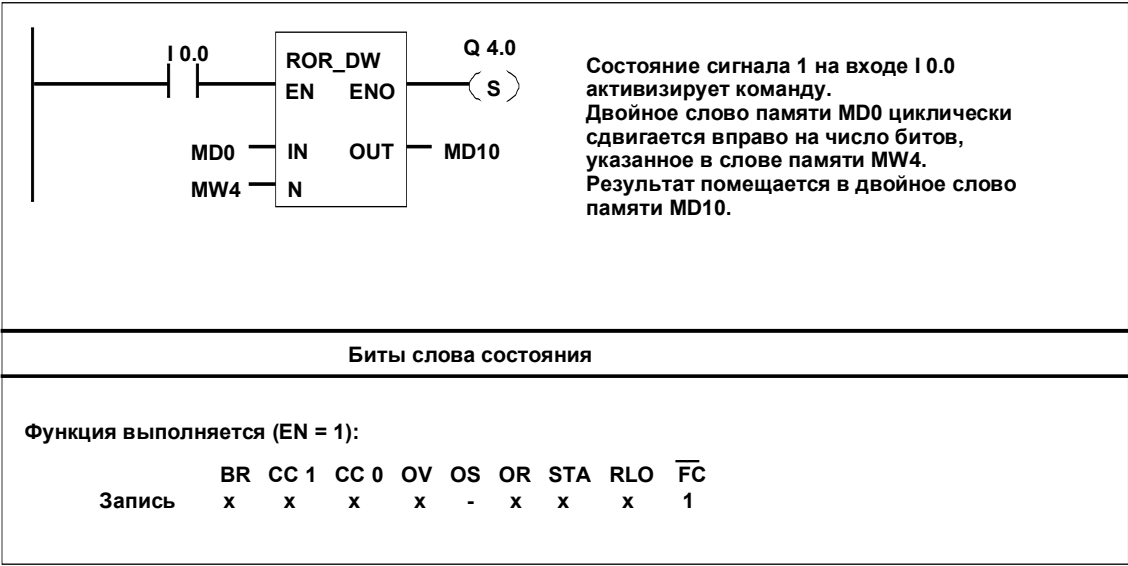


Рис. 12–12 Циклический сдвиг вправо двойного слова

13 Операции с блоками данных

Обзор главы

Раздел	Описание	Страница
13.1	Открытие блока данных: DB или DI	13–2

13.1 Открытие блока данных: DB или DI

Описание

Вы можете использовать команду *Открыть блок данных: DB или DI*, чтобы открыть уже существующий блок данных как DB или DI. Номер блока данных передается в регистр DB или DI. Последующие команды DB и DI обращаются к соответствующим блокам в зависимости от содержимого регистра.

Таблица 13–1. Элемент “Открытие блока данных: DB или DI” и параметры, с указанием международного краткого имени

Элемент KOP	Параметр	Тип данных	Область памяти	Описание
<Номер DB> или <Номер DI> ——(OPN)	Номер DB или DI	BLOCK_DB	-	Диапазон номеров DB или DI зависит от вашего CPU.

Таблица 13–2. Элемент “Открытие блока данных: DB или DI” и параметры, с указанием сокращенного имени SIMATIC

Элемент KOP	Параметр	Тип данных	Область памяти	Описание
<Номер DB> или <Номер DI> ——(AUF)	Номер DB или DI	BLOCK_DB	-	Диапазон номеров DB или DI зависит от вашего CPU.

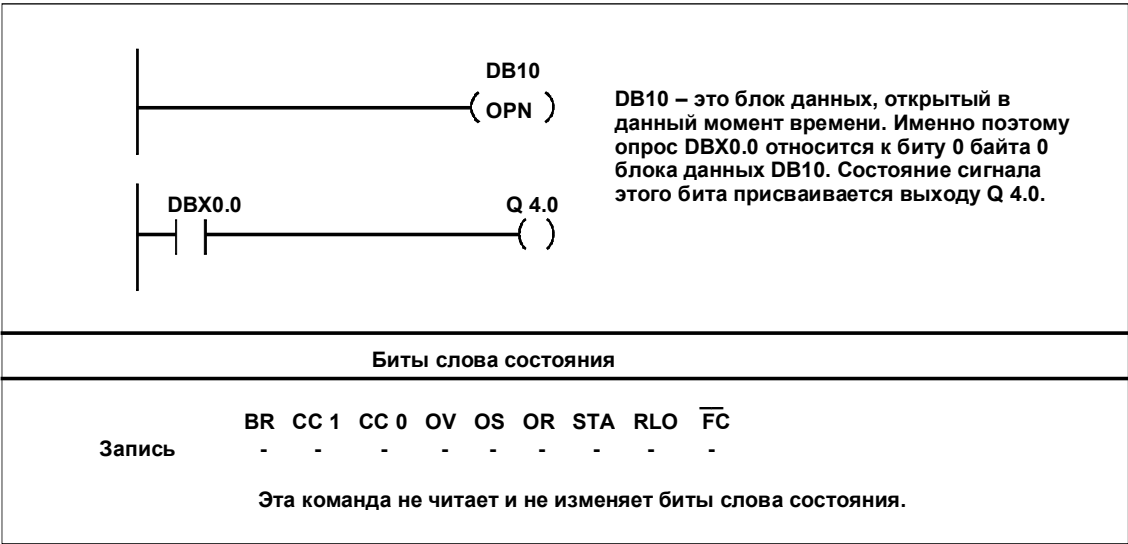


Рис. 13–1. Открытие блока данных: DB или DI

14 Команды перехода

Обзор главы

Раздел	Описание	Стр.
14.1	Обзор	14–2
14.2	Переход в блоке, если RLO = 1 (безусловный переход)	14–3
14.3	Переход в блоке, если RLO = 1 (условный переход)	14–4
14.4	Переход в блоке, если RLO = 0 (переход, если не 1)	14–5
14.5	Метка	14–6

14.1 Обзор

Метка как адрес

Метка является адресом команды перехода. Метка состоит максимум из четырех символов. Первый символ должен быть буквой алфавита; остальные символы могут быть буквами или цифрами (например, SEG3). Метка перехода указывает пункт назначения, в который по вашему желанию должна перейти программа.

Метку вводят над катушкой перехода (см. рис. 14–1).

Метка как пункт назначения

Метка пункта назначения должна располагаться в начале сегмента. Метку пункта назначения вводят в начале сегмента, выбирая LABEL из браузера KOP. Появляется пустой блок. В этом блоке вводится имя метки (см. рис. 14–1).

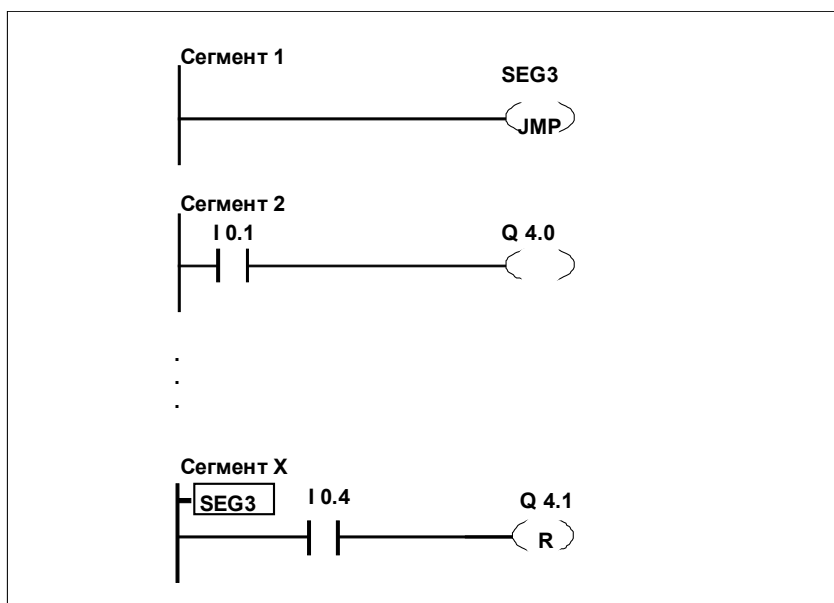


Рис. 14–1. Метка как адрес и пункт назначения

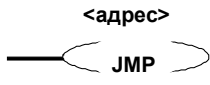
14.2 Переход в блоке, если RLO = 1 (безусловный переход)

Описание

Команда *Безусловный переход* соответствует команде «перейти на метку». Между левой шиной и операцией не могут располагаться никакие дополнительные элементы КОР. Ни одна из команд между операцией перехода и меткой не выполняется.

Вы можете использовать эту команду во всех логических блоках: организационных блоках (OB), функциональных блоках (FB) и функциях (FC).

Таблица 14–1. Элемент “Безусловный переход” и параметры

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	Имя метки	-	-	Адрес определяет метку, на которую выполняется абсолютный переход.

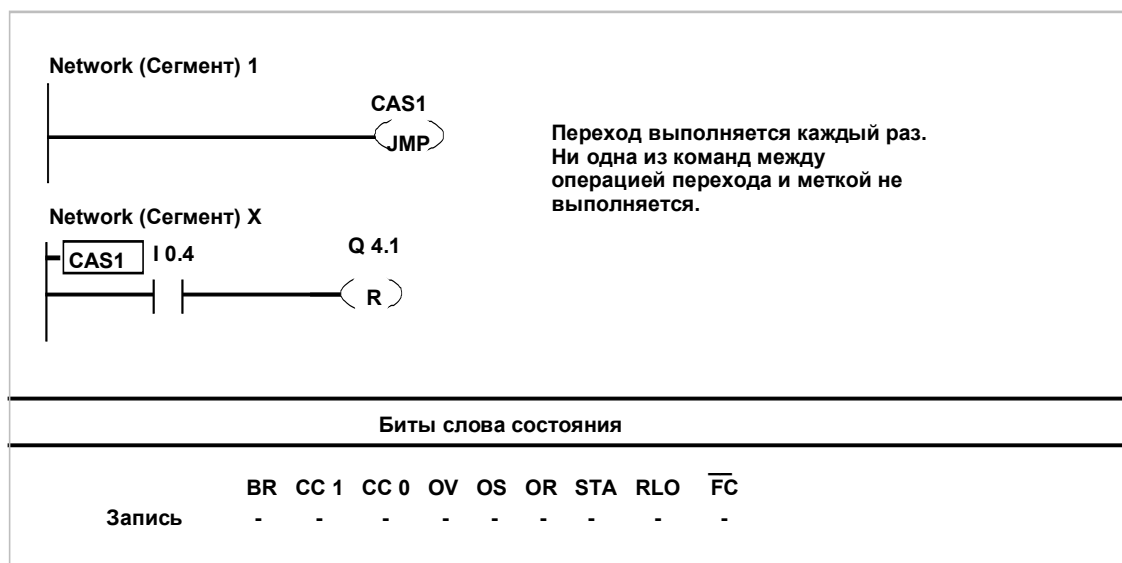



Рис. 14–2. Безусловный переход: переход на метку

14.3 Переход в блоке, если RLO = 1 (условный переход)

Описание

Команда *Условный переход* соответствует команде «перейти на метку», если RLO = 1. Для этой операции используйте элемент KOP "Jump unconditional [Безусловный переход]", но только с предшествующей логической операцией. Условный переход выполняется только тогда, когда результатом этой логической операции является RLO = 1. Ни одна из команд между операцией перехода и меткой не выполняется.

Вы можете использовать эту команду во всех логических блоках: организационных блоках (OB), функциональных блоках (FB) и функциях (FC).

Таблица 14–2. Элемент “Условный переход” и параметры				
Элемент KOP	Параметр	Тип данных	Область памяти	Описание
<div><адрес> </div>	Имя метки	-	-	Адрес определяет метку, на которую выполняется переход, когда RLO = 1.

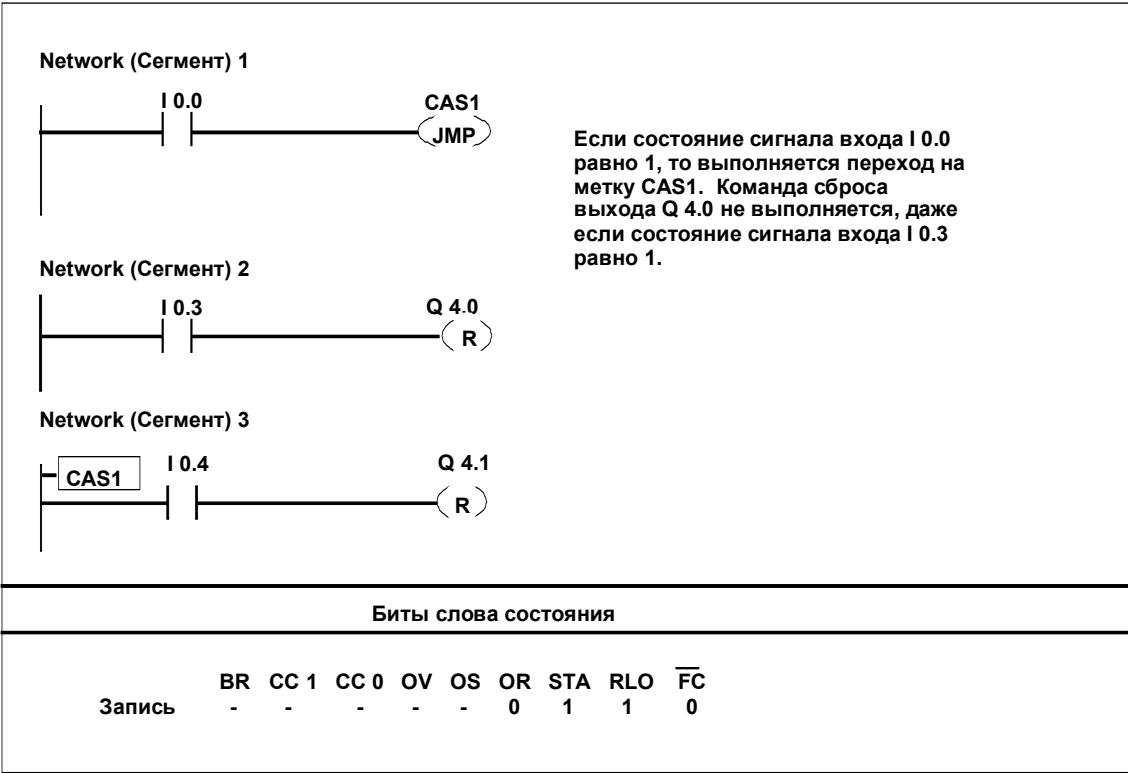


Рис. 14–3. Условный переход: переход на метку


14.4 Переход в блоке, если RLO = 0 (переход, если не 1)

Описание

Команда *Переход, если не 1* соответствует команде «перейти на метку», которая выполняется, если RLO = 0.

Вы можете использовать эту команду во всех логических блоках: организационных блоках (OB), функциональных блоках (FB) и функциях (FC).

Таблица 14–3. Элемент “Переход, если не 1” и параметры

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	Имя метки	-	-	Адрес определяет метку, на которую происходит переход, когда RLO = 0.

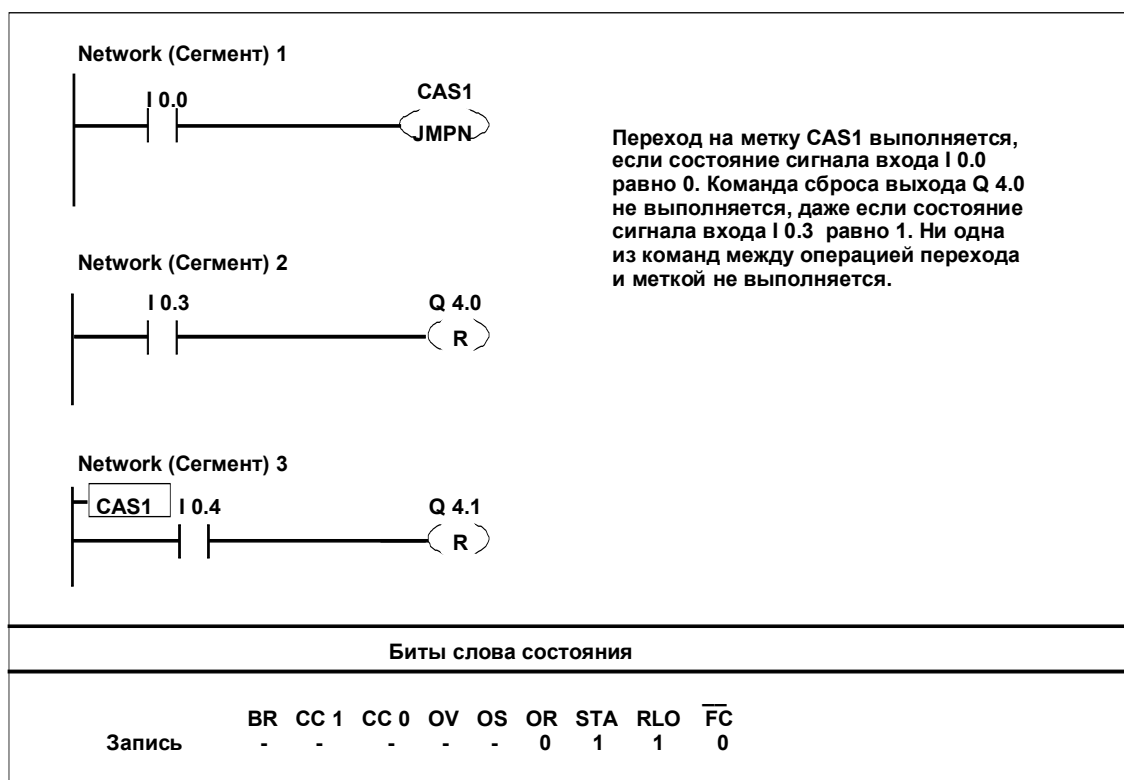



Рис. 14–4. Переход, если не 1

14.5 Метка

Описание

LABEL [метка] – это идентификатор пункта назначения команды перехода.
Для каждой из команд ---(JMP) и ---(JMPN) должна существовать метка.

Элемент КОР	Описание
	4 символа: Первый символ: буква остальные символы: буква или цифра

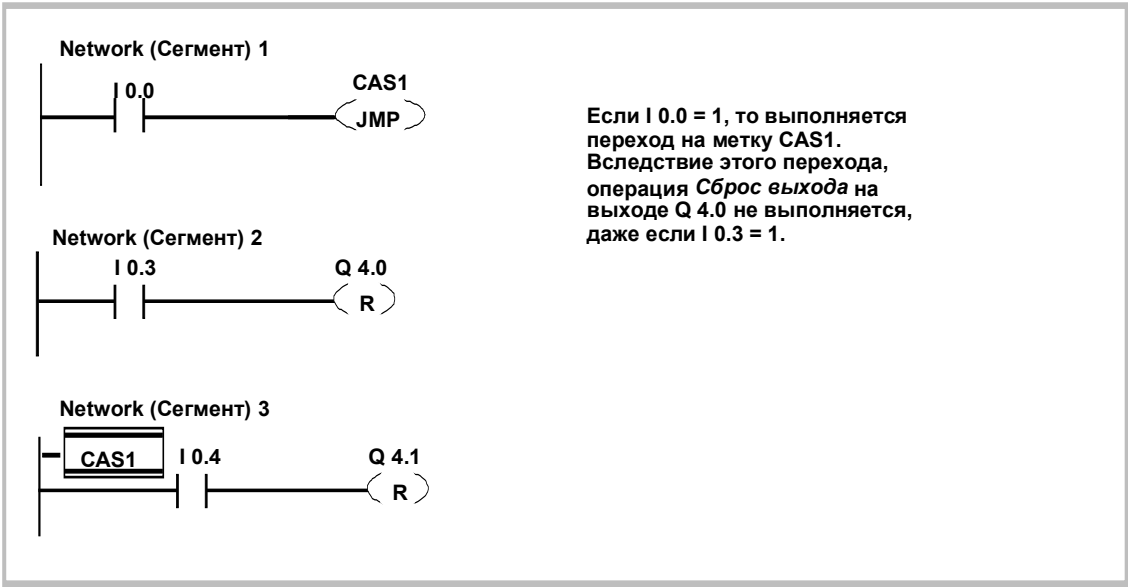


Рис. 14–5. Метка

15 Операции с битами состояния

Обзор главы

Раздел	Описание	Стр.
15.1	Обзор	15–2
15.2	Бит ошибки "Регистр BR"	15–3
15.3	Биты результата	15–4
15.4	Бит ошибки "Недопустимая операция"	15–6
15.5	Бит ошибки "Переполнение"	15–7
15.6	Бит ошибки "Сохраняемое переполнение"	15–8

15.1 Обзор

Описание

Операции с битами состояния – это битовые логические операции (см. раздел 4.1), которые работают с битами слова состояния (см. раздел 2.3). Каждая из этих операций реагирует на одно из следующих условий, отображаемых одним или несколькими битами слова состояния:

- Установлен бит двоичного результата (то есть он имеет состояние сигнала 1).
- Результат арифметической операции относится к 0 одним из следующих способов:
 - больше нуля (>0)
 - меньше нуля (<0)
 - больше или равен нулю (≥ 0)
 - меньше или равен нулю (≤ 0)
 - равен нулю ($=0$)
 - не равен нулю ($\neq 0$)
- Результат арифметической операции является недопустимым.
- В арифметической операции возникло переполнение.

В последовательном соединении операции с битами состояния логически связывают результат своего опроса состояния сигнала с предыдущим результатом логической операции в соответствии с таблицей истинности операции И (см. раздел 2.2 и таблицу 2–8). В параллельном соединении операции с битами состояния логически связывают свой результат с предыдущим RLO в соответствии с таблицей истинности операции ИЛИ (см. раздел 2.2 и таблицу 2–9).

В этой главе элемент *Бит ошибки "Регистр BR"*, опрашивающий состояние сигнала бита BR (Binary Result [двоичный результат]) слова состояния, показан в международной форме и в форме SIMATIC.

Слово состояния

Слово состояния – это регистр в памяти вашего CPU, содержащий биты, к которым вы можете обращаться в операндах логических операций над битами и словами. Рис. 15–1 показывает структуру слова состояния. Дополнительную информацию по отдельным битам слова состояния вы найдете в разделе 2.3.

2^{15} 2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	\overline{FC}

Рис. 15–1. Структура слова состояния

Параметры

Следующие элементы KOP не обладают вводимыми параметрами.

15.2 Бит ошибки "Регистр BR"

Описание

Вы можете использовать команду *Бит ошибки "Регистр BR"* для опроса состояния сигнала бита BR (Binary Result [двоичный результат]) слова состояния (см. раздел 2.3). При использовании в последовательном соединении эта команда комбинирует результат своего опроса с предыдущим результатом логической операции (RLO) в соответствии с таблицей истинности логической функции И (см. раздел 2.2 и таблицу 2–8). При использовании в параллельном соединении эта команда комбинирует результат своего опроса с предыдущим RLO в соответствии с таблицей истинности логической функции ИЛИ (см. раздел 2.2 и таблицу 2–9).

Элемент и его отрицательная форма

Рис. 15–2 показывает элемент *Бит ошибки "Регистр BR"* и его отрицательную форму. Эти элементы изображены с их международным сокращенным именем и сокращенным именем SIMATIC.

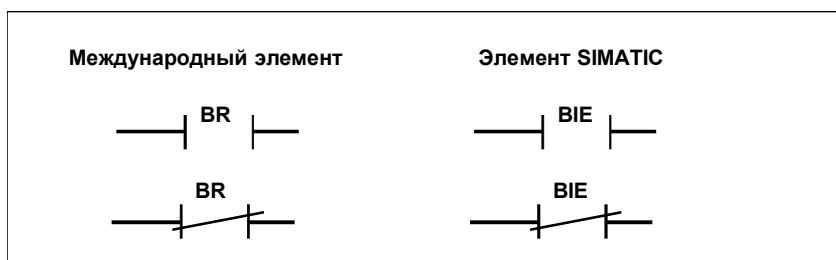


Рис. 15–2. Элемент Бит ошибки "Регистр BR" и его отрицательная форма

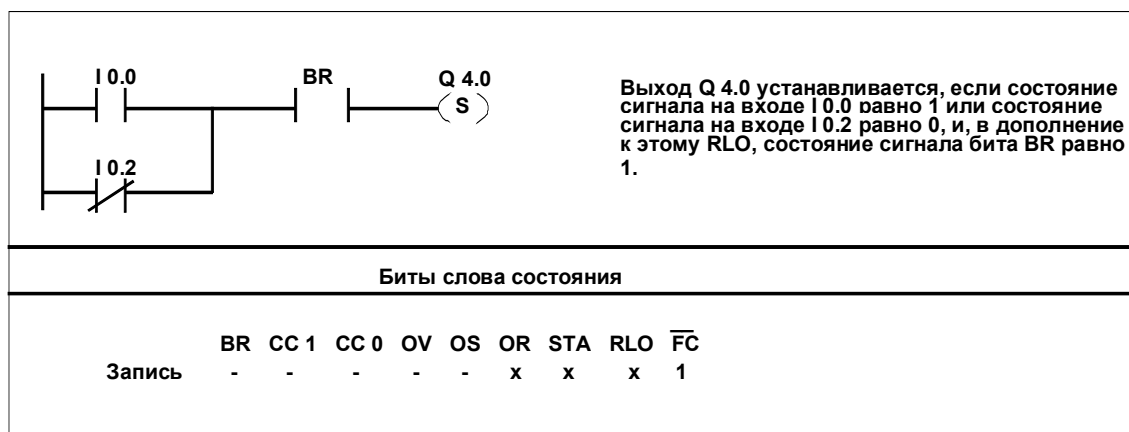


Рис. 15–3. Бит ошибки "Регистр BR"

15.3 Биты результата

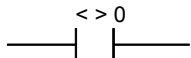
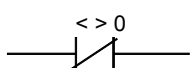
Описание

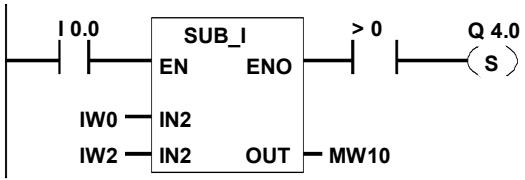
Вы можете использовать команды *Бит результата* для определения отношения результата математической операции к нулю: >0 , <0 , ≥ 0 , ≤ 0 , $=0$ или $\neq 0$. Эта команда использует сравнение с нулем как свой операнд (см. таблицу 15–1). Внутренне CPU обращается к битам кода условия слова состояния (CC 1 и CC 0, см. раздел 2.3) и опрашивает комбинацию состояний сигнала в этих ячейках. Эта комбинация сообщает CPU об отношении результата к 0. Если условие сравнения, отраженное в этом операнде, выполняется, то результат опроса состояния сигнала равен 1.

При использовании в последовательном соединении эта команда комбинирует результат своего опроса с предыдущим результатом логической операции (RLO) в соответствии с таблицей истинности функции И (см. раздел 2.2 и таблицу 2–8). При использовании в параллельном соединении эта команда комбинирует результат своего опроса с предыдущим RLO в соответствии с таблицей истинности функции ИЛИ (см. раздел 2.2 и таблицу 2–9).

Таблица 15–1. Элементы "Бит результата" и их отрицательные формы

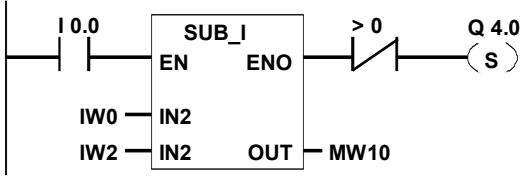
Элемент КОР	Описание
> 0 	Команда <i>Бит результата "Больше 0"</i> определяет, действительно ли результат математической операции больше 0. Эта команда опрашивает комбинацию битов CC 1 и CC 0 (коды условия) слова состояния, чтобы определить отношение результата к 0.
< 0 	Команда <i>Бит результата "Меньше 0"</i> определяет, действительно ли результат математической операции меньше 0. Эта команда опрашивает комбинацию битов CC 1 и CC 0 (коды условия) слова состояния, чтобы определить отношение результата к 0.
≥ 0 	Команда <i>Бит результата "Больше или равно 0"</i> определяет, действительно ли результат математической операции больше или равен 0. Эта команда опрашивает комбинацию битов CC 1 и CC 0 (коды условия) слова состояния, чтобы определить отношение результата к 0.
≤ 0 	Команда <i>Бит результата "Меньше или равно 0"</i> определяет, действительно ли результат математической операции меньше или равен 0. Эта команда опрашивает комбинацию битов CC 1 и CC 0 (коды условия) слова состояния, чтобы определить отношение результата к 0.
$= 0$ 	Команда <i>Бит результата "Равно 0"</i> определяет, действительно ли результат математической операции равен 0. Эта команда опрашивает комбинацию битов CC 1 и CC 0 (коды условия) слова состояния, чтобы определить отношение результата к 0.

Таблица 15–1. Элементы "Бит результата" и их отрицательные формы	
Элемент КОР	Описание
	Команда <i>Бит результата "Не равно 0"</i> определяет, действительно ли результат математической операции не равен 0. Эта команда опрашивает комбинацию битов CC 1 и CC 0 (коды условия) слова состояния, чтобы определить отношение результата к 0..
	



Если состояние сигнала на входе I 0.0 равно 1, то блок SUB_I активизируется. Если значение входного слова IW0 больше, чем значение входного слова IW2, то результат математической операции IW0 - IW2 больше 0. Если состояние сигнала EN равно 1 (активизировано) и во время выполнения команды произошла ошибка, то состояние сигнала ENO равно 0.

Выход Q 4.0 устанавливается, если функция выполняется должным образом и результат больше 0. Если состояние сигнала на входе I 0.0 равно 0 (не активизирован), то состояния сигналов EN и ENO равны 0.



Выход Q 4.0 устанавливается, если функция выполняется должным образом и результат меньше или равен 0. Если состояние сигнала на входе I 0.0 равно 0 (не активизирован), то состояния сигналов EN и ENO равны 0. Если состояние сигнала EN равно 1 (активизирован) и при выполнении команды произошла ошибка, то состояние сигнала ENO равно 0.

Биты слова состояния									
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	-	-	-	-	-	x	x	x	1

Рис. 15–4. Бит результата "Больше 0" и инвертированный бит результата "Больше 0"

15.4 Бит ошибки "Недопустимая операция"

Описание

Вы можете использовать команду *Бит ошибки "Недопустимая операция"* для проверки того, является ли результат математической операции над числами с плавающей точкой допустимым (т.е. не является ли какое-либо из значений в математической функции недопустимым числом с плавающей точкой). Для этого оцениваются биты кодов условия слова состояния (CC 1 и CC 0, см. раздел 2.3). Если результат математической операции недопустим (unordered, UO), то опрос состояния сигнала дает результат 1. Если комбинация битов CC 1 и CC 0 не дает информации о недопустимости, то результат опроса состояния сигнала равен 0.

При использовании в последовательном соединении эта команда комбинирует результат своего опроса с предыдущим результатом логической операции (RLO) в соответствии с таблицей истинности функции И (см. раздел 2.2 и таблицу 2–8). При использовании в параллельном соединении эта команда комбинирует результат своего опроса с предыдущим RLO в соответствии с таблицей истинности функции ИЛИ (см. раздел 2.2 и таблицу 2–9).

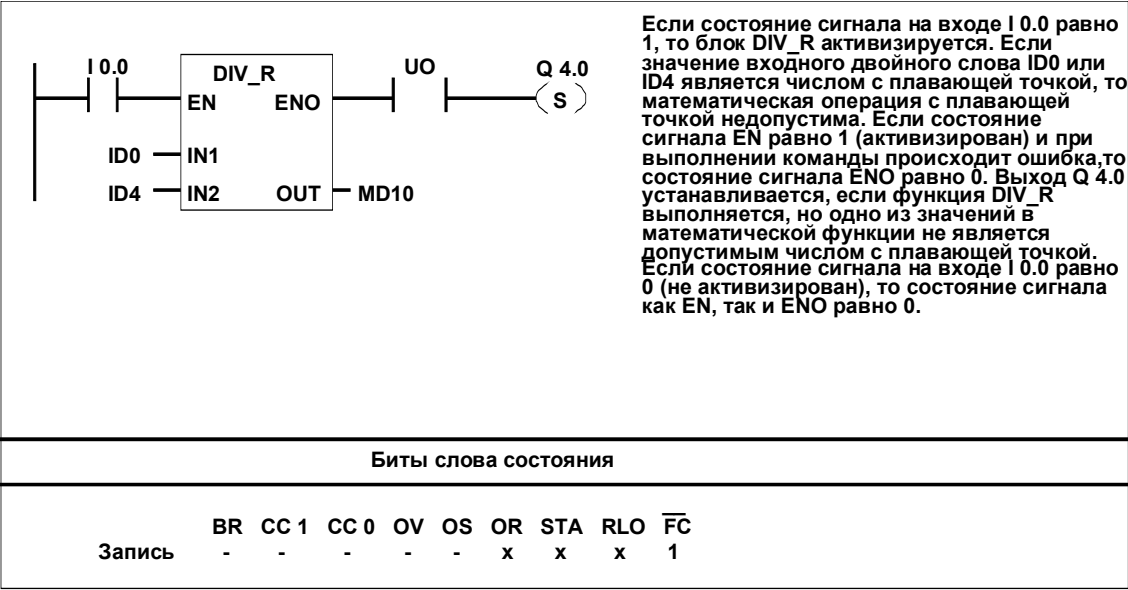


Рис. 15–5. Бит ошибки "Недопустимая операция"

15.5 Бит ошибки "Переполнение"

Описание

Вы можете использовать команду *Бит ошибки "Переполнение"* для распознавания переполнения (overflow, OV) в последней математической операции. Если после выполнения системой последней математической операции результат выходит за пределы допустимого отрицательного или положительного диапазона, то бит OV в слове состояния (см. раздел 2.3) устанавливается. Команда опрашивает состояние этого бита. Этот бит сбрасывается математической операцией, выполняемой без ошибок.

При использовании в последовательном соединении эта команда комбинирует результат своего опроса с предыдущим результатом логической операции (RLO) в соответствии с таблицей истинности функции И (см. раздел 2.2 и таблицу 2–8). При использовании в параллельном соединении эта команда комбинирует результат своего опроса с предыдущим RLO в соответствии с таблицей истинности функции ИЛИ (см. раздел 2.2 и таблицу 2–9).

Network [Сегмент] 1:

Network [Сегмент] 2:

Если состояние сигнала на входе I 0.0 равно 1, то блок SUB_I активизируется. Если результат математической операции входное слово IW0 минус входное слово IW2 выходит за пределы допустимого диапазона для целых чисел, то бит OV в слове состояния устанавливается. Результат опроса состояния сигнала для OV равен 1. Выход Q 4.0 устанавливается, если опрос для OV равен 1 и RLO сегмента 2 равен 1 (т.е., если RLO непосредственно перед выходом Q 4.0 равен 1).

Если состояние сигнала на входе I 0.0 равно 0 (не активизирован), то состояние сигнала как на EN, так и на ENO равно 0. Если состояние сигнала EN равно 1 (активизирован) и результат математической операции выходит за пределы допустимого диапазона, то состояние сигнала ENO равно 0.

Замечание: Опрос OV необходим только в случае различных сегментов. В противном случае можно брать выход ENO математической операции, который равен 0, если результат выходит за пределы допустимого диапазона.

Биты слова состояния									
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Запись	-	-	-	-	-	x	x	x	1

Рис. 15–6. Бит ошибки "Переполнение"

15.6 Бит ошибки "Сохраняемое переполнение"

Описание

Вы можете использовать команду *Бит ошибки "Сохраняемое переполнение"* для распознавания переполнения с фиксацией (сохраняемого переполнения = overflow stored, OS) в математической операции. Если после выполнения системой математической операции результат выходит за пределы допустимого отрицательного или положительного диапазона, то бит OS в слове состояния (см. раздел 2.3) устанавливается. Команда опрашивает состояние этого бита. В отличие от бита OV (переполнение) бит OS остается установленным при безошибочном выполнении математических операций (см. раздел 15.5).

При использовании в последовательном соединении эта команда комбинирует результат своего опроса с предыдущим результатом логической операции (RLO) в соответствии с таблицей истинности функции И (см. раздел 2.2 и таблицу 2–8). При использовании в параллельном соединении эта команда комбинирует результат своего опроса с предыдущим RLO в соответствии с таблицей истинности функции ИЛИ (см. раздел 2.2 и таблицу 2–9).

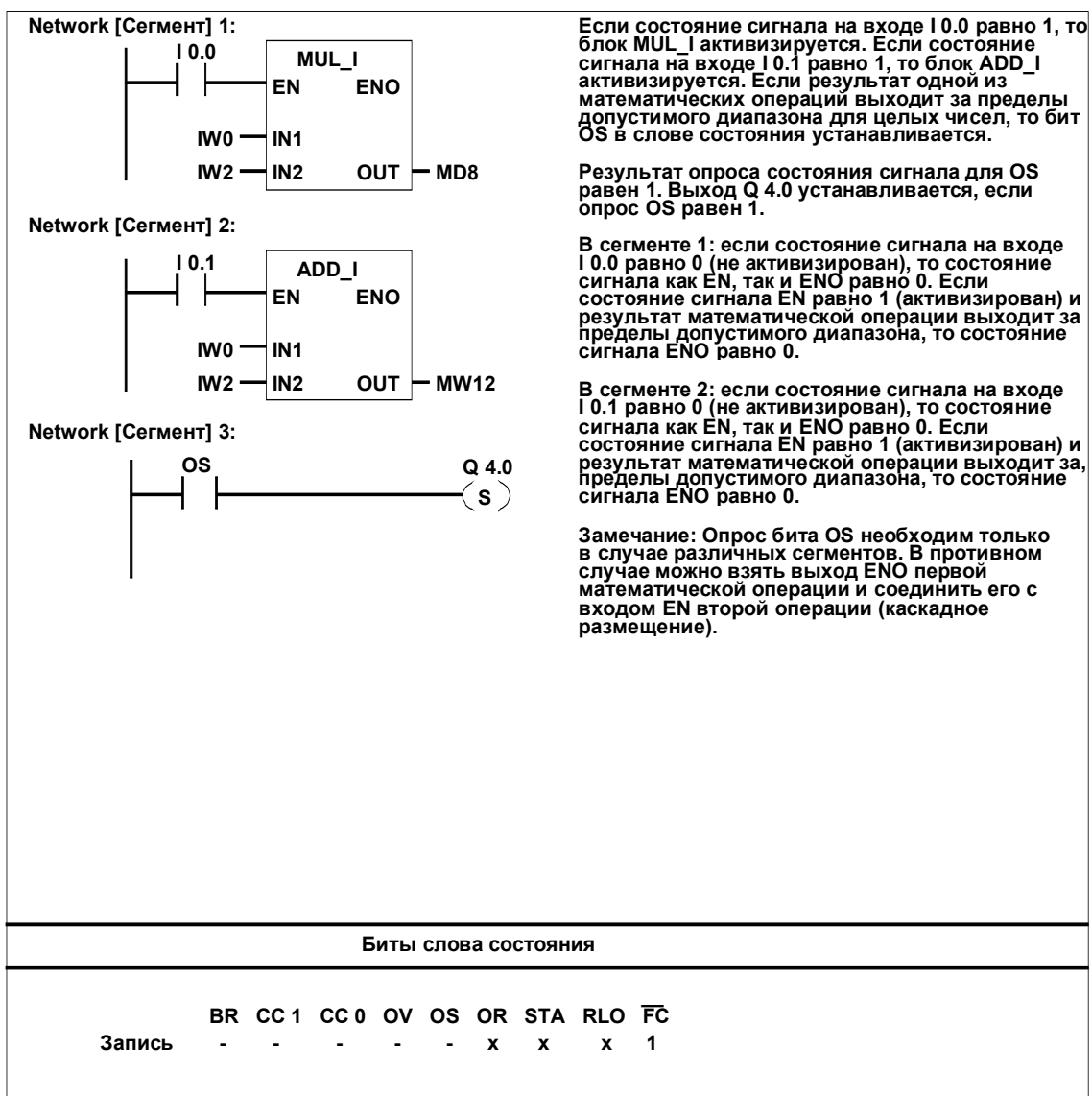


Рис. 15–7. Бит ошибки "Сохраняемое переполнение"

16 Команды управления программой

Обзор главы

Раздел	Описание	Стр.
16.1	Вызов FC/SFC из катушки	16–2
16.2	Вызов FB, FC, SFB, SFC и мультиэкземпляров	16–4
16.3	Возврат	16–7
16.4	Команды главного управляющего реле	16–9
16.5	Активизация/деактивизация главного управляющего реле	16–10
16.6	Включение/выключение главного управляющего реле	16–13

16.1 Вызов FC/SFC из катушки

Описание

Вы можете использовать команду *Вызвать FC/SFC из катушки* для того, чтобы вызвать функцию (FC) или системную функцию (SFC), не имеющую параметров. В зависимости от предшествующей связи вызов является условным или безусловным (см. пример на рис. 16–1).

В случае условного вызова вы не можете в разделе кода функции (FC) вводить параметры типа BLOCK_FC. Однако в пределах функционального блока (FB), вы можете вводить BLOCK_FC как тип параметра.

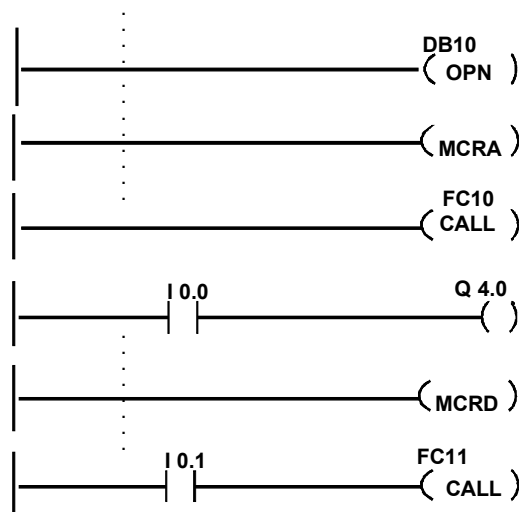
Условный вызов выполняется только тогда, когда RLO равен 1. Если условный вызов не выполняется, то RLO после команды вызова равен 0. Если команда выполняется, то она выполняет следующие функции:

- Сохраняет адрес, в котором нуждается, чтобы возвратиться в вызывающий блок
- Сохраняет селекторы обоих текущих блоков данных (DB и DI)
- Заменяет текущую область локальных данных на предыдущую область локальных данных
- Помещает бит MA (MCR Active bit [бит *MCR активно*]) в стек блоков (BSTACK)
- Создает новую область локальных данных для вызываемой FC или SFC

После всего этого обработка программы продолжается в вызванном блоке. Информацию о передаче параметров смотрите в *Оперативной справке STEP 7*.

Таблица 16–1. Элемент “Вызов FC/SFC из катушки” и параметры

Элемент KOP	Параметр	Тип данных	Область памяти	Описание
	Номер	BLOCK_FC	-	<p>Номер FC или SFC (например, FC10 или SFC59). Набор доступных SFC зависит от вашего CPU.</p> <p>В случае условного вызова вы не можете внутри функции (FC) вводить параметры типа BLOCK_FC. Однако внутри функционального блока вы можете вводить BLOCK_FC как тип параметра.</p>



Если выполняется безусловный вызов FC10, то команда CALL выполняет следующие функции:
 Сохраняет адрес, в котором нуждается для того, чтобы возвратиться в текущий FB.
 Сохраняет селекторы для DB10 и для экземплярного блока данных FB.
 Помещает бит MA, установленный в 1 в команде MCRA, в стек блоков (BSTACK) и сбрасывает этот бит в 0 для вызываемого FC10.
 Обработка программы продолжается в FC10. Если вы хотите использовать функцию MCR в FC10, то вы должны снова активизировать ее там. Когда FC10 завершается, обработка программы возвращается в вызывающий FB..
 Бит MA восстанавливается, и DB10 и экземплярный блок данных определяемого пользователем FB снова являются текущими DB, независимо от того, какие DB использовал FC10.
 После возврата из FC10 состояние сигнала входа I 0.0 присваивается выходу Q 4.0. Вызов FC11 – это условный вызов. Он выполняется только тогда, когда состояние сигнала входа I 0.0 равно 1. Если вызов выполняется, то реализуются такие же функции, как при вызове FC10.

Биты слова состояния

Безусловный вызов

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	\overline{FC}
Запись	-	-	-	-	0	0	1	-	0

Условный вызов

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	\overline{FC}
Запись	-	-	-	-	0	0	1	1	0

Рис. 16–1. Вызов FC/SFC из катушки

16.2 Вызов FB, FC, SFB, SFC и мультитекземпляров

Описание

Вы можете вызывать функциональные блоки (FB), функции (FC), системные функциональные блоки (SFB), системные функции (SFC) и мультитекземпляры, выбирая их из списка "Program Elements [Элементы программы]". Они находятся в конце списка семейств команд под следующими названиями:

- FB Blocks [Блоки FB]
- FC Blocks [Блоки FC]
- SFB Blocks [Блоки SFB]
- SFC Blocks [Блоки SFC]
- Multiple Instances [Мультитекземпляры]
- Libraries [Библиотеки]

Когда вы выбираете один из этих блоков, на вашем экране появляется прямоугольник с номером или символическим именем функции или функционального блока и принадлежащими ему параметрами.

Вызываемый вами блок должен быть скомпилирован и уже существовать в вашем программном файле, библиотеке или в CPU.

Если команда вызова FB, FC, SFB, SFC и мультитекземпляров выполняется, то она выполняет следующие функции:

- Сохраняет адрес, в котором нуждается, чтобы возвратиться в вызывающий блок
- Сохраняет селекторы обоих текущих блоков данных (DB и DI)
- Заменяет предыдущую область локальных данных на текущую область локальных данных
- Помещает бит MA (MCR Active bit [бит *MCR активно*]) в стек блоков (BSTACK)
- Создает новую область локальных данных для вызываемой FC или SFC

Примечание

Когда сохраняются регистры DB и DI, они не могут указывать на блоки данных, которые вы открыли. Из-за механизма передачи входных и выходных параметров, особенно там, где дело касается функциональных блоков, компилятор иногда перезаписывает регистр DB. Подробности сморите в *Оперативной справке STEP 7*.

После этого обработка программы продолжается в вызванном блоке.

Разрешающий выход

Разрешающий выход (ENO) блока KOP соответствует биту BR слова состояния (см. раздел 2.3). При записи функционального блока или функции, которую вы хотите вызывать из KOP, независимо от того, записываете ли вы FB или FC в форме AWL или KOP, вы несете ответственность за управление битом BR. Вы должны использовать команду SAVE (в AWL) или катушку --- (SAVE) (в KOP), чтобы сохранить RLO в бите BR в соответствии со следующими критериями:

- Сохранить RLO со значением 1 в бите BR для случая, когда FB или FC выполняется без ошибки.
- Сохранить RLO со значением 0 в бите BR для случая, когда FB или FC выполняется с ошибкой.

Вы должны программировать эти команды в конце FB или FC так, чтобы они были последними командами, которые выполняются в блоке.



Предупреждение

Возможен непреднамеренный сброс бита BR в 0.

Если при записи FB и FC в форме KOP вы обрабатываете бит BR не так, как описано выше, то один FB или FC может перезаписать бит BR другого FB или FC.

Чтобы избежать этой проблемы, сохраняйте RLO в конце каждого FB или FC так, как описано выше.

Влияние вызова на биты слова состояния

Рис. 16–2 показывает влияние условного и безусловного вызова блока на биты слова состояния (см. раздел 2.3).

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	\overline{FC}
Условный:	Запись	x	-	-	-	0	0	1	1	x
Безусловный:	Запись	-	-	-	-	0	0	1	-	x

Рис. 16–2. Влияние вызова блока на биты слова состояния

Параметры

Параметры, определенные в разделе VAR блока, будут отображаться в блоке KOP. Снабжение блока параметрами различается в зависимости от типа блока следующим образом:

- В случае функции (FC) вы должны предоставить фактические параметры для всех формальных параметров
- В случае функциональных блоков (FB) ввод фактических параметров необязателен. Однако вы должны закрепить за FB экземплярный блок данных (экземплярный DB). Если формальному параметру не поставлен в соответствие фактический параметр, то FB работает со значениями, существующими в его экземплярном DB.
- В случае мультиэкземпляров вам не нужно указывать экземплярный DB, так как вызываемому блоку уже назначен номер DB (за дополнительной информацией об описании мультиэкземпляров обращайтесь к *Оперативной справке STEP 7*).

Структурным параметрам IN/OUT и параметрам типа «Pointer [указатель]» и типа «Array [массив]» вы должны предоставить в распоряжение фактический параметр (по крайней мере, во время первого вызова).

Каждый фактический параметр, который вы предоставляете в распоряжение при вызове функционального блока, должен иметь такой же тип данных, как его формальный параметр.

Информацию о том, как программировать функцию или как работать с ее параметрами, вы можете получить в *Оперативной справке STEP 7*.

Таблица 16-2 показывает блок для вызова FB, FC, SFB, SFC и мультиэкземпляров и описывает общие для всех этих блоков параметры блока вызова. Когда вы вызываете свой блок из браузера команд, в верхней части блока автоматически появляется номер блока (номер FB, FC, SFB или SFC, например, FC10).

Таблица 16–2. Блок и параметры вызова FB, FC, SFB, SFC и мультиэкземпляров				
Блок КОР	Параметр	Тип данных	Область памяти	Описание
<div>DB No. <div>Block no. EN ENO IN OUT IN/OUT</div></div>	DB No.	BLOCK_DB	-	Номер экземплярного блока данных. Вам нужно вводить эту информацию только для вызова FB.
	EN	BOOL	I, Q, M, D, L	Разрешающий вход
	ENO	BOOL	I, Q, M, D, L	Разрешающий выход

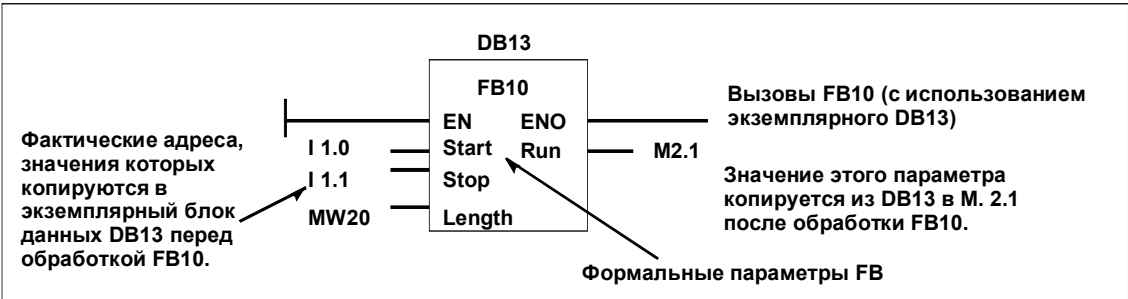


Рис. 16–3. Вызов FB в виде блока КОР

16.3 Возврат

Описание

Вы можете использовать команду *Возврат* для выхода из блоков. Вы можете выходить из блока по условию. Возврат сохраняет RLO в бите BR слова состояния.

Если выход из блока происходит вследствие условного возврата, то состояние сигнала битов RLO и BR в блоке, которому возвращается управление программой, равно 1.

Таблица 16–3. Элемент “Возврат”

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

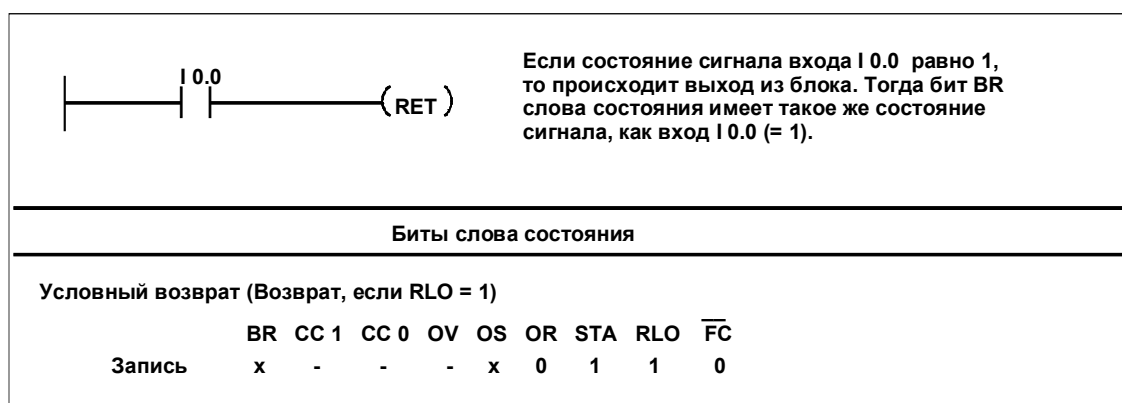


Рис. 16–4. Возврат

Важные замечания относительно использования функций MCR

Будьте осторожны с блоками, в которых главное управляющее реле было активизировано посредством MCRA:

- Если MCR деактивизировано, то всеми присваиваниями в сегментах программы между (MCR<) и (MCR>) записывается значение 0.
- MCR деактивизируется, если RLO был =0 перед командой (MCR<).



Опасность

ПЛК находится в состоянии STOP или имеют место неопределенные характеристики во время выполнения!

Для вычисления адресов компилятор использует также доступ на запись к локальным данным, которые скрываются за временными переменными, определенными в VAR_TEMP.

Обращение к формальным параметрам

- Обращение к компонентам сложных параметров FC типа STRUCT, UDT, ARRAY, STRING.
- Обращение к компонентам сложных параметров FB типа STRUCT, UDT, ARRAY, STRING из области IN_OUT в блоке версии 2.
- Обращение к параметрам функционального блока версии 2, если его адрес больше, чем 8180.0.
- Обращение в функциональном блоке версии 2 к параметру типа BLOCK_DB открывает DB0. Любой последующий доступ к данным переводит CPU в режим STOP. T 0, C 0, FC0 или FB0 всегда используются для TIMER, COUNTER, BLOCK_FC и BLOCK_FB.

Передача параметров

- Вызовы, в которых передаются параметры.

KOP/FUP

- Т-ветви и промежуточные выходы (коннекторы) в KOP или FUP, запускаемые с RLO=0.

Исправление:

Устраните зависимость вышеупомянутых команд от MCR:

1. Деактивизируйте главное управляющее реле при помощи команды *Деактивизировать главное управляющее реле* перед рассматриваемым оператором или сегментом.
2. Активизируйте снова главное управляющее реле при помощи команды *Активизировать главное управляющее реле* после рассматриваемого оператора или сегмента.

16.4 Команды главного управляющего реле

Определение главного управляющего реле

Главное управляющее реле (Master Control Relay, MCR, см. также раздел 16.5) – это главный выключатель Американской системы электроавтоматики (контактного плана, КОР) для подачи питания и прерывания потока энергии (пути тока). Прерванный путь тока соответствует последовательности команд, которая записывает нулевое значение вместо расчетного значения, или последовательности команд, которая оставляет существующее значение в памяти неизменным. Операции, запускаемые командами, показанными в таблице 16–4, зависят от MCR.

Команды *Выходная катушка* и *Промежуточный выход (коннектор)* записывают в память 0, если MCR равен 0. Команды *Установить выход* и *Сбросить выход* оставляют существующее значение неизменным (см. таблицу 16–5).

Таблица 16–4. Команды, на которые влияет зона MCR

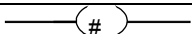
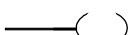
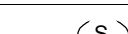

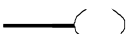

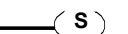
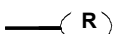
Элемент или имя в блоке	Имя команды	Раздел в этом руководстве
	Промежуточный выход (коннектор)	4.5
	Выходная катушка	4.4
	Установка выхода	4.8
	Сброс выхода	4.9
SR	Триггер "Установка-сброс", SR-триггер	4.22
RS	Триггер "Сброс-установка", RS-триггер	4.23
MOVE	Присвоить значение	10.1

Таблица 16–5. Операции, зависящие от MCR, и их реакция на его состояние сигнала


Состояние сигнала MCR	Выходная катушка или промежуточный выход  	Установка или сброс выхода   SR RS	Присвоить значение MOVE
0	Записывает 0 (Имитирует реле, которое при снятии напряжения переходит в свое состояние покоя)	Не записывает (Имитирует реле с механической блокировкой, которое при снятии напряжения остается в своем текущем состоянии)	Записывает 0 (Имитирует компонент, который при потере напряжения, выдает значение 0)
1	Нормальное выполнение	Нормальное выполнение	Нормальное выполнение

16.5 Активизация/деактивизация главного управляющего реле

Активизация MCR

При помощи команды *Активизировать главное управляющее реле* вы включаете зависимость последующих команд от MCR. После ввода этой команды вы можете программировать зоны MCR с этими командами (см. раздел 16.6). Когда ваша программа активизирует область MCR, все действия MCR зависят от содержимого стека MCR (см. рис. 16–6).


Таблица 16–6. Элемент “Активизация главного управляющего реле”

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

Деактивизация MCR

При помощи команды *Деактивизировать главное управляющее реле* вы выключаете зависимость последующих команд от MCR. После этой команды вы больше не можете программировать зоны MCR. Когда ваша программа деактивизирует область MCR, MCR всегда пропускает ток, независимо от записей в стеке MCR.

Таблица 16–7. Элемент “Деактивизация главного управляющего реле”

Элемент KOP	Параметр	Тип данных	Область памяти	Описание
	Нет	-	-	Деактивизирует функцию MCR

Стек MCR и бит, управляющий зависимостью от него (бит МА), относятся к соответствующему уровню и должны сохраняться и выбираться каждый раз, когда переключается уровень последовательности команд. Они предварительно устанавливаются в начале каждого уровня последовательности (биты входов MCR с 1 по 8 устанавливаются в 1, указатель вершины стека MCR устанавливается в 0, и бит МА устанавливается в 0).

Стек MCR передается от блока к блоку, а бит МА сохраняется и устанавливается в 0, каждый раз, когда блок вызывается. Он извлекается обратно в конце блока.

MCR может быть реализован таким способом, что он оптимизирует продолжительность работы CPU, генерирующих код. Причиной этого является то, что зависимость от MCR не передается блоком дальше; она должна явно активизироваться командой MCR. CPU, генерирующий код, распознает эту команду и генерирует дополнительный код, необходимый для оценивания стека MCR до тех пор, пока он не распознает команду MCR или не достигнет конца блока. В случае команд, лежащих вне области MCRA/MCRD, нет увеличения времени выполнения.

Команды MCRA и MCRD должны использоваться в пределах вашей программы всегда попарно.

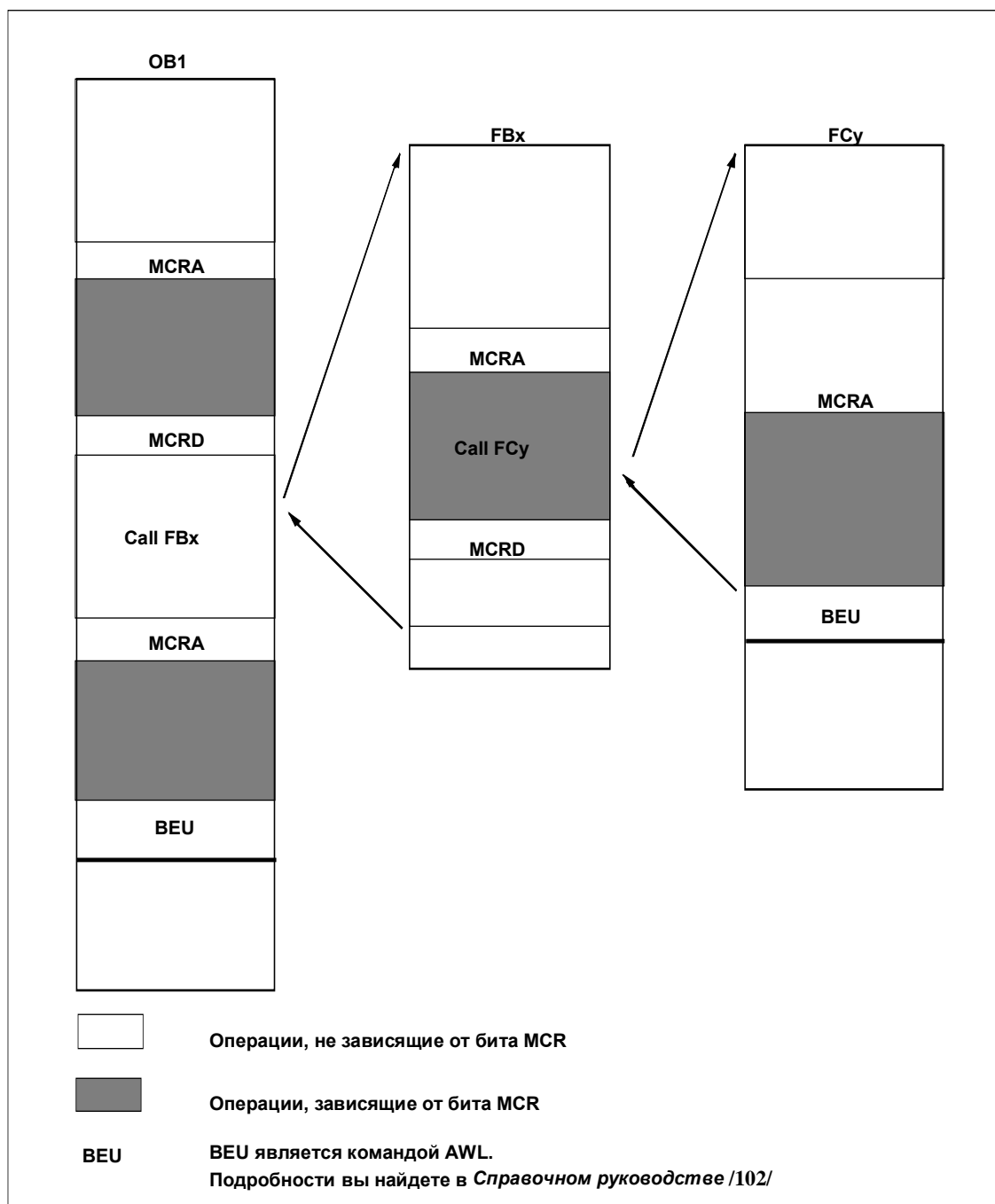


Рис. 16–5. Активизация и деактивизация области MCR

Операции, запрограммированные между MCRA и MCRD, зависят от состояния сигнала бита MCR. Операции, запрограммированные вне последовательности MCRA–MCRD, не зависят от состояния сигнала бита MCR. Если команда MCRD отсутствует, то операции, запрограммированные между командами MCRA и BEU, зависят от бита MCR. (BEU – это команда AWL. Дополнительную информацию вы найдете в *Руководстве /232/*.)

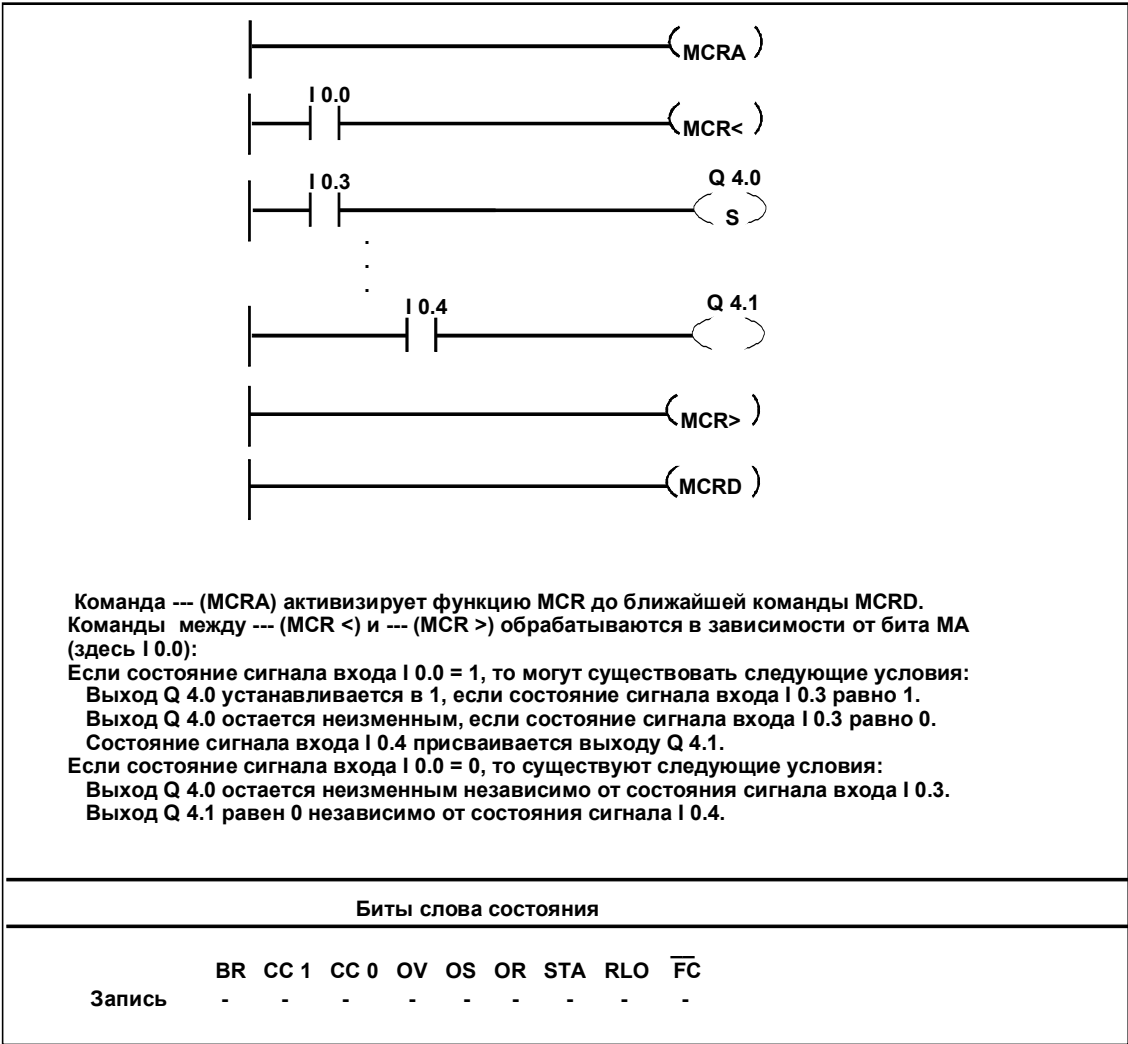


Рис. 16–6. Главное управляющее реле (активизация и деактивизация)

Вы должны самостоятельно программировать в блоках зависимость функций (FC) и функциональных блоков (FB) от MCR. Если эта функция или этот функциональный блок вызывается из последовательности MCRA/MCRD, то не все команды в пределах этой последовательности автоматически зависят от бита MCR. Чтобы добиться этого, используйте команду MCRA вызываемого блока.



Предупреждение


Риск травмы персонала и повреждения оборудования:
Никогда не используйте команду MCR в качестве АВАРИЙНОГО ВЫКЛЮЧАТЕЛЯ] или устройства защиты персонала.
MCR не является заменой аппаратного главного управляющего реле.

16.6 Включение/выключение главного управляющего реле

Включение MCR

Команда *Включить главное управляющее реле* (MCR<) запускает операцию, которая помещает RLO в стек MCR и открывает зону MCR. На команды, показанные в таблице 16-4, влияет RLO, который помещается в стек MCR, когда открывается зона MCR. Стек MCR работает по принципу LIFO (Last In, First Out [последним вошел – первым вышел]). Возможны только восемь записей. Если стек уже заполнен, то команда *Включить главное управляющее реле* выдает ошибку стека MCR (MCRF).


Таблица 16–8. Элемент “Включение главного управляющего реле”

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

Выключение MCR

Команда *Выключить главное управляющее реле* (MCR>) закрывает зону MCR, которая была открыта последней. Команда делает это, удаляя запись RLO из стека MCR. RLO был помещен туда командой *Включить главное управляющее реле*. Ячейка, освободившаяся на другом конце стека MCR, работающего по принципу LIFO (Last In, First Out [последним вошел – первым вышел]), устанавливается в 1. Если стек уже пуст, то команда *Выключить главное управляющее реле* выдает ошибку стека MCR (MCRF).

Таблица 16–9. Элемент “Выключение главного управляющего реле”

Элемент КОР	Параметр	Тип данных	Область памяти	Описание
	Нет	-	-	Закрывает зону MCR, которая была открыта последней

Управление MCR происходит посредством стека шириной в один бит и глубиной восемь записей (см. рис. 16–7). MCR активизирован до тех пор, пока все восемь записей в стеке не станут равными 1. Команда --(MCR <) копирует RLO в стек MCR. Команда --(MCR >) удаляет последнюю запись из стека и устанавливает освободившуюся ячейку стека в 1. Если происходит ошибка, например, более восьми команд --(MCR >) следуют за друг другом или вы пытаетесь выполнить команду --(MCR >), когда стек пуст, то такая ошибка активизирует сообщение об ошибке MCRF. Состояние стека MCR контролируется по указателю вершины стека (MSP: 0 = пустой, 1 = одна запись, 2 = две записи, ..., 8 = восемь записей).

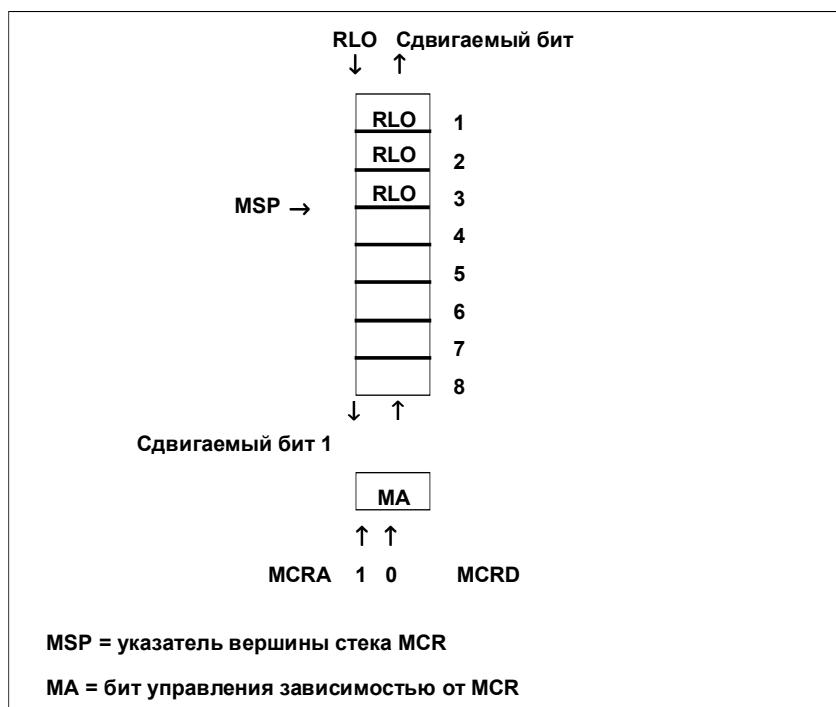


Рис. 16–7. Стек главного управляющего реле

Команды `--(MCR<)` и `--(MCR>)` должны использоваться в пределах вашей программы всегда попарно.

Команда `--(MCR<)` принимает состояние сигнала RLO и копирует его в бит MCR.

Команда `--(MCR>)` устанавливает бит MCR в 1 абсолютно. Из-за этого свойства любая другая команда между командами `--(MCRA)` и `--(MCRD)` работает независимо от бита MCR (информацию об `--(MCRA)` и `--(MCRD)` смотрите выше).

Вложение команд `(MCR<)` и `(MCR>)`

Вы можете вкладывать пары команд `--(MCR<)` и `--(MCR>)` друг в друга. Максимальная глубина вложения равна восьми, то есть вы можете записать максимум восемь команд `--(MCR<)` одну за другой перед тем, как вставить команду `--(MCR>)`. Вы должны программировать одинаковое количество команд `--(MCR<)` и `--(MCR>)`.

Если команды `--(MCR<)` вкладываются друг в друга, то формируется бит MCR более низкого уровня вложенности. Тогда команда `--(MCR<)` связывает текущий RLO с текущим битом MCR в соответствии с таблицей истинности логической операции И.

Когда команда `--(MCR>)` завершает уровень вложенности, она извлекает бит MCR из следующего более высокого уровня.

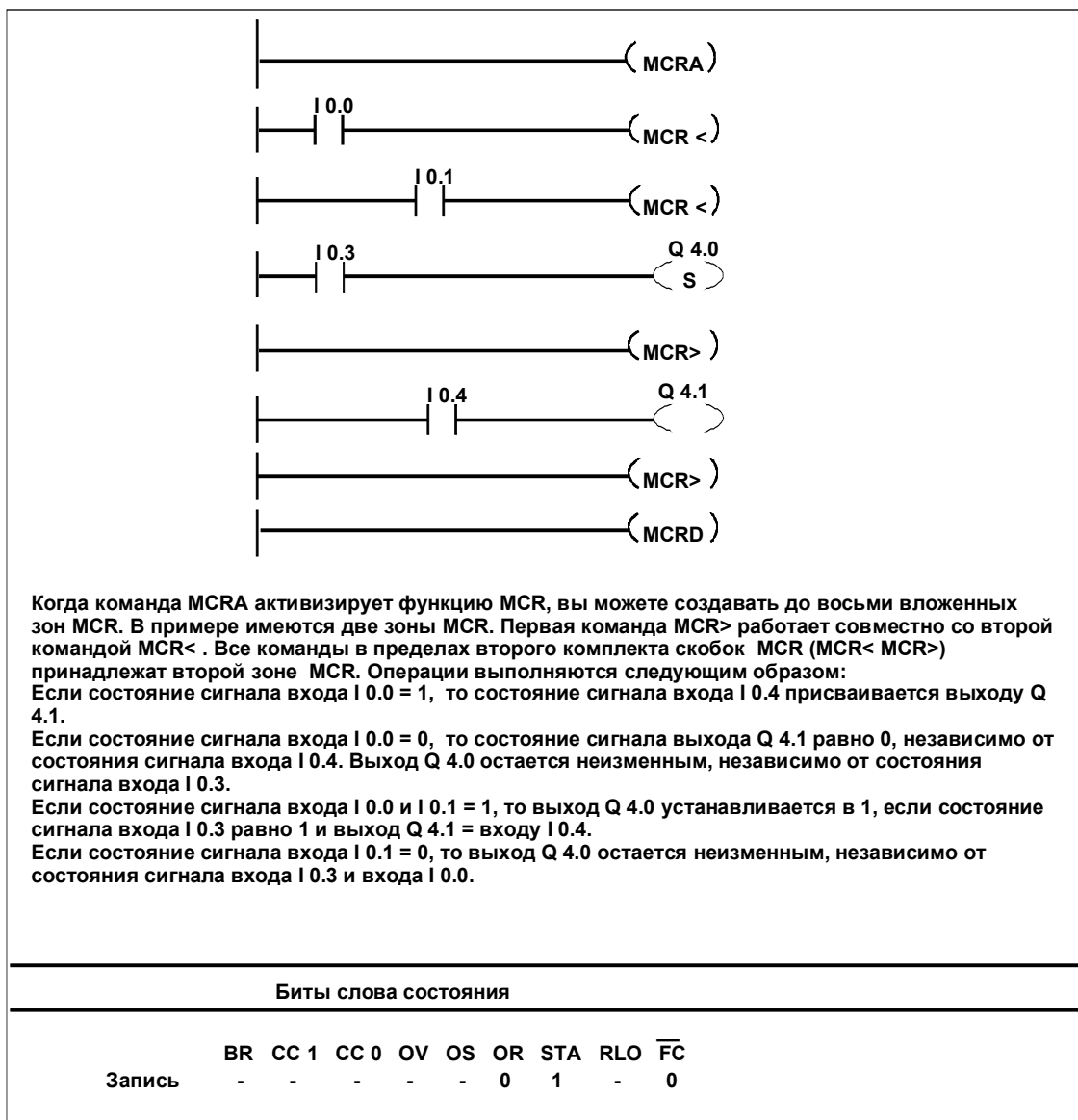


Рис. 16–8. Выключение главного управляющего реле

А Алфавитный список команд

Обзор главы

Раздел	Описание	Стр.
A.1	Список международных наименований	A–2
A.2	Список международных наименований с их русскими соответствиями	A–5
A.3	Список русских наименований	A–8
A.4	Список русских наименований и их международных соответствий	A–11
A.5	Список международных сокращенных наименований и сокращенных наименований SIMATIC	A–14

A.1 Список международных наименований

В таблице А–1 представлен в алфавитном порядке список команд с их полными международными именами. Вслед за каждым полным именем идет его международное сокращение и ссылка на страницу в данном руководстве, где эта команда объясняется.

Таблица А–1. Команды КОР (Ladder Logic), упорядоченные по алфавиту в соответствии с их полными международными именами, и с сокращениями

Полное имя	Сокращение	№ стр.
Add Double Integer	ADD_DI	7–3
Add Integer	ADD_I	7–2
Add Real	ADD_R	8–3
Address Negative Edge Detection	NEG	4–23
Address Positive Edge Detection	POS	4–22
Assign a Value	MOVE	10–2
BCD to Double Integer	BCD_DI	10–7
BCD to Integer	BCD_I	10–4
Call FB from Box	CALL_FB	16–4
Call FC from Box	CALL_FC	16–4
Call FC SFC from Coil (without parameters)	----(CALL)	16–2
Call System FB from Box	CALL_SFB	16–4
Call System FC from Box	CALL_SFC	16–4
Ceiling	CEIL	10–17
Compare Double Integer (>, <, ==, <>, <=, >=)	CMP>=D	9–3
Compare Integer (>, <, ==, <>, <=, >=)	CMP>=I	9–2
Compare Real (>, <, ==, <>, <=, >=)	CMP>=R	9–5
Divide Double Integer	DIV_DI	7–9
Divide Integer	DIV_I	7–8
Divide Real	DIV_R	8–6
Double Integer to BCD	DI_BCD	10–8
Double Integer to Real	DI_R	10–9
Down Counter	S_CD	6–8
Down Counter Coil	----(CD)	4–14
Exception Bit BR Memory	BR --- ---	15–4
Exception Bit Overflow	OV --- ---	15–8
Exception Bit Overflow Stored	OS --- ---	15–10
Exception Bit Unordered	UO --- ---	15–7
Extended Pulse S5 Timer	S_PEXT	5–8
Extended Pulse Timer Coil	--(SE)	4–16
Floor	FLOOR	10–18
Integer to BCD	I_BCD	10–5
Integer to Double Integer	I_DI	10–6
Invert Power Flow	--- NOT ---	4–8
Jump–If–Not	---(JMPN)	14–5
Jump	---(JMP)	14–3
Master Control Relay Activate	---(MCRA)	16–11
Master Control Relay Deactivate	---(MCRD)	16–11
Master Control Relay Off	---(MCR>)	16–14
Master Control Relay On	---(MCR<)	16–14
Midline Output	---(#)--	4–7
Multiply Double Integer	MUL_DI	7–7
Multiply Integer	MUL_I	7–6
Multiply Real	MUL_R	8–5

Таблица А-1. Команды КОР (Ladder Logic), упорядоченные по алфавиту в соответствии с их полными международными именами, и с сокращениями

Полное имя	Сокращение	№ стр.
Negate Real Number	NEG_R	10-14
Negated Exception Bit BR Memory	BR --- / ---	15-4
Negated Exception Bit Overflow	OV --- / ---	15-8
Negated Exception Bit Overflow Stored	OS --- / ---	15-10
Negated Exception Bit Unordered	UO --- / ---	15-7
Negated Result Bit Equal 0	==0 --- / ---	15-5
Negated Result Bit Greater Equal 0	>=0 --- / ---	15-5
Negated Result Bit Greater Than 0	>0 --- / ---	15-5
Negated Result Bit Less Equal 0	<=0 --- / ---	15-5
Negated Result Bit Less Than 0	<0 --- / ---	15-5
Negated Result Bit Not Equal 0	<>0 --- / ---	15-5
Negative RLO Edge Detection	---(N)---	4-21
Normally Closed Contact (Address)	--- / ---	4-4
Normally Open Contact (Address)	--- ---	4-3
Off-Delay S5 Timer	S_OFFDT	5-15
Off-Delay Timer Coil	---(SF)	4-19
On-Delay S5 Timer	S_ODT	5-10
On-Delay Timer Coil	---(SD)	4-17
ONEs Complement Double Integer	INV_DI	10-11
ONEs Complement Integer	INV_I	10-10
Open Data Block: DB or DI	---(OPN)	13-2
Output Coil	---()	4-5
Positive RLO Edge Detection	---(P)---	4-20
Pulse S5 Timer	S_PULSE	5-6
Pulse Timer Coil	---(SP)	4-15
Reset Coil	---(R)	4-11
Reset-Set Flipflop	RS	4-25
Result Bit Equal 0	==0 --- ---	15-5
Result Bit Greater Equal 0	>=0 --- ---	15-5
Result Bit Greater Than 0	>0 --- ---	15-5
Result Bit Less Equal 0	<=0 --- ---	15-5
Result Bit Less Than 0	<0 --- ---	15-5
Result Bit Not Equal 0	<>0 --- ---	15-5
Retentive On-Delay S5 Timer	S_ODTS	5-13
Retentive On-Delay Timer Coil	---(SS)	4-18
Return	---(RET)	16-8
Return Fraction Double Integer	MOD	7-10
Rotate Left Double Word	ROL_DW	12-10
Rotate Right Double Word	ROR_DW	12-12
Round to Double Integer	ROUND	10-15
Save RLO to BR Memory	---(SAVE)	4-9
Set Coil	---(S)	4-10
Set Counter Value	---(SC)	4-12
Set-Reset Flipflop	SR	4-24
Shift Left Double Word	SHL_DW	12-4
Shift Left Word	SHL_W	12-2
Shift Right Double Integer	SHR_DI	12-9
Shift Right Double Word	SHR_DW	12-6
Shift Right Integer	SHR_I	12-7
Shift Right Word	SHR_W	12-5
Subtract Double Integer	SUB_DI	7-5
Subtract Integer	SUB_I	7-4
Subtract Real	SUB_R	8-4

Таблица А–1. Команды КОР (Ladder Logic), упорядоченные по алфавиту в соответствии с их полными международными именами, и с сокращениями

Полное имя	Сокращение	№ стр.
Truncate Double Integer Part	TRUNC	10–16
TWOs Complement Double Integer	NEG_DI	10–13
TWOs Complement Integer	NEG_I	10–12
Up Counter	S_CU	6–6
Up Counter Coil	---(CU)	4–13
Up–Down Counter	S_CUD	6–4
(Word) And Double Word	WAND_DW	11–5
(Word) And Word	WAND_W	11–3
(Word) Exclusive Or Double Word	WXOR_DW	11–13
(Word) Exclusive Or Word	WXOR_W	11–11
(Word) Or Double Word	WOR_DW	11–9
(Word) Or Word	WOR_W	11–7

A.2 Список международных наименований с их русскими соответствиями

В таблице A–2 представлен в алфавитном порядке список команд с их полными международными именами. Вслед за каждым полным именем идет его русский эквивалент и ссылка на страницу в данном руководстве, где эта команда объясняется.

Таблица A–2. Команды KOP (Ladder Logic), упорядоченные по алфавиту в соответствии с их полными международными именами, с русскими соответствиями

Международное имя	Русское соответствие	№ стр.
Add Double Integer	Сложение двойных целых чисел	7–3
Add Integer	Сложение целых чисел	7–2
Add Real	Сложение вещественных чисел	8–3
Address Negative Edge Detection	Обнаружение отрицательного фронта сигнала	4–23
Address Positive Edge Detection	Обнаружение положительного фронта сигнала	4–22
Assign a Value	Присвоить значение	10–2
BCD to Double Integer	Преобразовать двоично-десятичное число в двойное целое	10–7
BCD to Integer	Преобразовать двоично-десятичное число в целое	10–4
Call FB from Box	Вызвать FB из окна списка	16–4
Call FC from Box	Вызвать FC из окна списка	16–4
Call FC SFC from Coil (without parameters)	Вызвать FC/SFC из катушки (без параметров)	16–2
Call System FB from Box	Вызвать системный FB из окна списка	16–4
Call System FC from Box	Вызвать системную FC из окна списка	16–4
Ceiling	Округлить до ближайшего большего целого числа	10–17
Compare Double Integer (>, <, ==, <>, <=, >=)	Сравнить двойные целые числа	9–3
Compare Integer (>, <, ==, <>, <=, >=)	Сравнить целые числа	9–2
Compare Real (>, <, ==, <>, <=, >=)	Сравнить вещественные числа	9–5
Divide Double Integer	Деление двойных целых чисел	7–9
Divide Integer	Деление целых чисел	7–8
Divide Real	Деление вещественных чисел	8–6
Double Integer to BCD	Преобразовать двойное целое число в двоично-десятичное	10–8
Double Integer to Real	Преобразовать двойное целое число в вещественное	10–9
Down Counter	Счетчик обратного счета	6–8
Down Counter Coil	Катушка со счетчиком обратного счета	4–14
Exception Bit BR Memory	Бит ошибки "Регистр BR"	15–4
Exception Bit Overflow	Бит ошибки "Переполнение"	15–8
Exception Bit Overflow Stored	Бит ошибки "Сохраняемое переполнение"	15–10
Exception Bit Unordered	Бит ошибки "Недопустимая операция"	15–7
Extended Pulse S5 Timer	Таймер S5 – формирователь удлиненного импульса (SV)	5–8
Extended Pulse Timer Coil	Катушка с таймером – формирователем удлиненного импульса (SV)	4–16
Floor	Округлить до ближайшего меньшего целого числа	10–18
Integer to BCD	Преобразовать целое число в двоично-десятичное	10–5
Integer to Double Integer	Преобразовать целое число в двойное целое	10–6
Invert Power Flow	Инvertировать результат логической операции (поток энергии)	4–8

Таблица А–2. Команды КОР (Ladder Logic), упорядоченные по алфавиту в соответствии с их полными международными именами, с русскими соответствиями

Международное имя	Русское соответствие	№ стр.
Jump–If–Not	Перейти, если не 1	14–5
Jump	Перейти (если 1)	14–3
Master Control Relay Activate	Активизировать главное управляющее реле	16–11
Master Control Relay Deactivate	Деактивизировать главное управляющее реле	16–11
Master Control Relay Off	Выключить главное управляющее реле	16–14
Master Control Relay On	Включить главное управляющее реле	16–14
Midline Output	Промежуточный выход (коннектор)	4–7
Multiply Double Integer	Умножение двойных целых чисел	7–7
Multiply Integer	Умножение целых чисел	7–6
Multiply Real	Умножение вещественных чисел	8–5
Negate Real Number	Изменить знак вещественного числа	10–14
Negated Exception Bit BR Memory	Инвертированный бит ошибки "Регистр BR"	15–4
Negated Exception Bit Overflow	Инвертированный бит ошибки "Переполнение"	15–8
Negated Exception Bit Overflow Stored	Инвертированный бит ошибки "Сохраняемое переполнение"	15–10
Negated Exception Bit Unordered	Инвертированный бит ошибки "Недопустимая операция"	15–7
Negated Result Bit Equal 0	Инвертированный бит результата "Равно 0"	15–5
Negated Result Bit Greater Equal 0	Инвертированный бит результата "Больше или равно 0"	15–5
Negated Result Bit Greater Than 0	Инвертированный бит результата "Больше 0"	15–5
Negated Result Bit Less Equal 0	Инвертированный бит результата "Меньше или равно 0"	15–5
Negated Result Bit Less Than 0	Инвертированный бит результата "Меньше 0"	15–5
Negated Result Bit Not Equal 0	Инвертированный бит результата "Не равно 0"	15–5
Negative RLO Edge Detection	Обнаружение отрицательного фронта RLO	4–21
Normally Closed Contact (Address)	Нормально замкнутый контакт (адрес)	4–4
Normally Open Contact (Address)	Нормально открытый контакт (адрес)	4–3
Off–Delay S5 Timer	Таймер S5 – формирователь задержки выключения (SA)	5–15
Off–Delay Timer Coil	Катушка с таймером – формирователем задержки выключения (SA)	4–19
On–Delay S5 Timer	Таймер S5 – формирователь задержки включения (SE)	5–10
On–Delay Timer Coil	Катушка с таймером – формирователем задержки включения (SE)	4–17
ONEs Complement Double Integer	Дополнение двойного целого числа до единицы	10–11
ONEs Complement Integer	Дополнение целого числа до единицы	10–10
Open Data Block: DB or DI	Открыть блок данных: DB или DI	13–2
Output Coil	Выходная катушка	4–5
Positive RLO Edge Detection	Обнаружение положительного фронта RLO	4–20
Pulse S5 Timer	Таймер S5 – формирователь импульса (SI)	5–6
Pulse Timer Coil	Катушка с таймером – формирователем импульса (SI)	4–15
Reset Coil	Сбросить выход	4–11
Reset–Set Flipflop	Сбросить-установить триггер, RS-триггер	4–25
Result Bit Equal 0	Бит результата "Равно 0"	15–5

Таблица А-2. Команды KOP (Ladder Logic), упорядоченные по алфавиту в соответствии с их полными международными именами, с русскими соответствиями

Международное имя	Русское соответствие	№ стр.
Result Bit Greater Equal 0	Бит результата "Больше или равно 0"	15-5
Result Bit Greater Than 0	Бит результата "Больше 0"	15-5
Result Bit Less Equal 0	Бит результата "Меньше или равно 0"	15-5
Result Bit Less Than 0	Бит результата "Меньше 0"	15-5
Result Bit Not Equal 0	Бит результата "Не равно 0"	15-5
Retentive On-Delay S5 Timer	Таймер S5 – формирователь задержки включения с запоминанием (SS)	5-13
Retentive On-Delay Timer Coil	Катушка с таймером – формирователем задержки включения с запоминанием (SS)	4-18
Return	Возврат	16-8
Return Fraction Double Integer	Получить остаток от деления двойного целого числа	7-10
Rotate Left Double Word	Выполнить циклический сдвиг двойного слова влево	12-10
Rotate Right Double Word	Выполнить циклический сдвиг двойного слова вправо	12-12
Round to Double Integer	Округлить до двойного целого числа	10-15
Save RLO to BR Memory	Сохранить RLO в регистре BR	4-9
Set Coil	Установить выход	4-10
Set Counter Value	Установить начальное значение счетчика	4-12
Set-Reset Flipflop	Установить-сбросить триггер, SR-триггер	4-24
Shift Left Double Word	Сдвинуть влево двойное слово	12-4
Shift Left Word	Сдвинуть влево слово	12-2
Shift Right Double Integer	Сдвинуть вправо двойное целое число	12-9
Shift Right Double Word	Сдвинуть вправо двойное слово	12-6
Shift Right Integer	Сдвинуть вправо целое число	12-7
Shift Right Word	Сдвинуть вправо слово	12-5
Subtract Double Integer	Вычитание двойных целых чисел	7-5
Subtract Integer	Вычитание целых чисел	7-4
Subtract Real	Вычитание вещественных чисел	8-4
Truncate Double Integer Part	Выделить целую часть числа	10-16
TWOs Complement Double Integer	Дополнение двойного целого числа до двух	10-13
TWOs Complement Integer	Дополнение целого числа до двух	10-12
Up Counter	Счетчик прямого счета	6-6
Up Counter Coil	Катушка со счетчиком прямого счета	4-13
Up-Down Counter	Счетчик прямого и обратного счета	6-4
(Word) And Double Word	Поразрядное И над двойными словами	11-5
(Word) And Word	Поразрядное И над словами	11-3
(Word) Exclusive Or Double Word	Поразрядное исключающее ИЛИ над двойными словами	11-13
(Word) Exclusive Or Word	Поразрядное исключающее ИЛИ над словами	11-11
(Word) Or Double Word	Поразрядное ИЛИ над двойными словами	11-9
(Word) Or Word	Поразрядное ИЛИ над словами	11-7

А.3 Список русских наименований

В таблице А–3 представлен в алфавитном порядке список команд с их полными именами на русском языке. Вслед за каждым полным именем идет его международное сокращенное имя и ссылка на страницу в данном руководстве, где эта команда объясняется.

Таблица А–3. Команды KOP (Ladder Logic), упорядоченные по алфавиту в соответствии с их русскими наименованиями, с краткими международными именами

Русское наименование	Сокращение	№ стр.
Активизировать главное управляющее реле	---(MCRA)	16–11
Бит ошибки "Недопустимая операция"	UO --- ---	15–7
Бит ошибки "Переполнение"	OV --- ---	15–8
Бит ошибки "Регистр BR"	BR --- ---	15–4
Бит ошибки "Сохраняемое переполнение"	OS --- ---	15–10
Бит результата "Больше 0"	>0 --- ---	15–5
Бит результата "Больше или равно 0"	>=0 --- ---	15–5
Бит результата "Меньше 0"	<0 --- ---	15–5
Бит результата "Меньше или равно 0"	<=0 --- ---	15–5
Бит результата "Не равно 0"	<>0 --- ---	15–5
Бит результата "Равно 0"	=0 --- ---	15–5
Включить главное управляющее реле	---(MCR<)	16–14
Возврат	---(RET)	16–8
Выключить главное управляющее реле	---(MCR>)	16–14
Выделить целую часть числа	TRUNC	10–16
Вызвать системную FC из окна списка	CALL_SFC	16–4
Вызвать системный FB из окна списка	CALL_SFB	16–4
Вызвать FB из окна списка	CALL_FB	16–4
Вызвать FC из окна списка	CALL_FC	16–4
Вызвать FC/SFC из катушки (без параметров)	----(CALL)	16–2
Выполнить циклический сдвиг двойного слова влево	ROL_DW	12–10
Выполнить циклический сдвиг двойного слова вправо	ROR_DW	12–10
Выходная катушка	---()	4–5
Вычитание вещественных чисел	SUB_R	8–4
Вычитание двойных целых чисел	SUB_DI	7–5
Вычитание целых чисел	SUB_I	7–4
Деактивизировать главное управляющее реле	---(MCRD)	16–11
Деление вещественных чисел	DIV_R	8–6
Деление двойных целых чисел	DIV_DI	7–9
Деление целых чисел	DIV_I	7–8
Дополнение двойного целого числа до двух	NEG_DI	10–13
Дополнение двойного целого числа до единицы	INV_DI	10–11
Дополнение целого числа до двух	NEG_I	10–12
Дополнение целого числа до единицы	INV_I	10–10
Изменить знак вещественного числа	NEG_R	10–14
Инвертированный бит ошибки "Недопустимая операция"	UO --- / ---	15–7
Инвертированный бит ошибки "Переполнение"	OV --- / ---	15–8
Инвертированный бит ошибки "Регистр BR"	BR --- / ---	15–4
Инвертированный бит ошибки "Сохраняемое переполнение"	OS --- / ---	15–10
Инвертированный бит результата "Больше 0"	>0 --- / ---	15–5
Инвертированный бит результата "Больше или равно 0"	>=0 --- / ---	15–5
Инвертированный бит результата "Меньше 0"	<0 --- / ---	15–5
Инвертированный бит результата "Меньше или равно 0"	<=0 --- / ---	15–5
Инвертированный бит результата "Не равно 0"	<>0 --- / ---	15–5
Инвертированный бит результата "Равно 0"	=0 --- / ---	15–5

Таблица А–3. Команды KOP (Ladder Logic), упорядоченные по алфавиту в соответствии с их русскими наименованиями, с краткими международными именами

Русское наименование	Сокращение	№ стр.
Инвертировать результат логической операции (поток энергии)	--- NOT ---	4–8
Катушка со счетчиком обратного счета	---(CD)	4–14
Катушка со счетчиком прямого счета	---(CU)	4–13
Катушка с таймером – формирователем задержки включения (SE)	---(SD)	4–17
Катушка с таймером – формирователем задержки включения с запоминанием (SS)	---(SS)	4–18
Катушка с таймером – формирователем задержки выключения (SA)	---(SF)	4–19
Катушка с таймером – формирователем импульса (SI)	---(SP)	4–15
Катушка с таймером – формирователем удлиненного импульса (SV)	---(SE)	4–16
Нормально замкнутый контакт	--- / ---	4–4
Нормально открытый контакт	--- ---	4–3
Обнаружение отрицательного фронта результата логической операции (RLO)	---(N)---	4–21
Обнаружение положительного фронта RLO	---(P)---	4–20
Обнаружение отрицательного фронта сигнала	NEG	4–23
Обнаружение положительного фронта сигнала	POS	4–22
Округлить до ближайшего большего целого числа	CEIL	10–17
Округлить до ближайшего меньшего целого числа	FLOOR	10–18
Округлить до двойного целого числа	ROUND	10–15
Открыть блок данных: DB или DI	---(OPN)	13–2
Перейти, если не 1	---(JMPN)	14–5
Перейти (если 1)	---(JMP)	14–3
Получить остаток от деления двойного целого числа	MOD	7–10
Поразрядное ИЛИ над двойными словами	WOR_DW	11–9
Поразрядное ИЛИ над словами	WOR_W	11–7
Поразрядное И над двойными словами	WAND_DW	11–5
Поразрядное И над словами	WAND_W	11–3
Поразрядное исключающее ИЛИ над двойными словами	WXOR_DW	11–13
Поразрядное исключающее ИЛИ над словами	WXOR_W	11–11
Преобразовать двоично-десятичное число в двойное целое	BCD_DI	10–7
Преобразовать двоично-десятичное число в целое	BCD_I	10–4
Преобразовать двойное целое число в вещественное	DI_R	10–9
Преобразовать двойное целое число в двоично-десятичное	DI_BCD	10–8
Преобразовать целое число в двоично-десятичное	I_BCD	10–5
Преобразовать целое число в двойное целое	I_DI	10–6
Присвоить значение	MOVE	10–2
Промежуточный выход (коннектор)	---(#)---	4–7
Сбросить выход	---(R)	4–11
Сбросить-установить триггер, RS-триггер	RS	4–25
Сдвинуть влево двойное слово	SHL_DW	12–4
Сдвинуть влево слово	SHL_W	12–2
Сдвинуть вправо двойное слово	SHR_DW	12–4
Сдвинуть вправо двойное целое число	SHR_DI	12–9
Сдвинуть вправо слово	SHR_W	12–5
Сдвинуть вправо целое число	SHR_I	12–7
Сложение вещественных чисел	ADD_R	8–3
Сложение двойных целых чисел	ADD_DI	7–3
Сложение целых чисел	ADD_I	7–2
Сохранить RLO в регистре BR	---(SAVE)	4–9
Сравнить вещественные числа	CMP_R_>=	9–5

Таблица А–3. Команды КОР (Ladder Logic), упорядоченные по алфавиту в соответствии с их русскими наименованиями, с краткими международными именами

Русское наименование	Сокращение	№ стр.
Сравнить двойные целые числа	CMP_D_>=	9–3
Сравнить целые числа	CMP_I_>=	9–2
Счетчик обратного счета	S_CD	6–8
Счетчик прямого и обратного счета	S_CUD	6–4
Счетчик прямого счета	S_CU	6–6
Таймер S5 – формирователь задержки включения (SE)	S_ODT	5–10
Таймер S5 – формирователь задержки включения с запоминанием (SS)	S_ODTS	5–13
Таймер S5 – формирователь задержки выключения (SA)	S_OFFDT	5–15
Таймер S5 – формирователь импульса (SI)	S_PULSE	5–6
Таймер S5 – формирователь удлиненного импульса (SV)	S_PEXT	5–8
Умножение вещественных чисел	MUL_R	8–5
Умножение двойных целых чисел	MUL_DI	7–7
Умножение целых чисел	MUL_I	7–6
Установить-сбросить триггер, SR-триггер	SR	4–24
Установить выход	---(S)	4–10
Установить начальное значение счетчика	---(SC)	4–12

А.4 Список русских наименований и их международных соответствий

В таблице А–4 представлен в алфавитном порядке список команд с их полными именами на русском языке. Вслед за каждым полным именем идет его международное соответствие и ссылка на страницу в данном руководстве, где эта команда объясняется.

Таблица А–4. Команды КОР (Ladder Logic), упорядоченные по алфавиту в соответствии с их русскими наименованиями, с их международными соответствиями

Русское наименование	Международное наименование	№ стр.
Активизировать главное управляющее реле	Master Control Relay Activate	16–11
Бит ошибки "Недопустимая операция"	Exception Bit Unordered	15–7
Бит ошибки "Переполнение"	Exception Bit Overflow	15–8
Бит ошибки "Регистр BR"	Exception Bit BR Memory	15–4
Бит ошибки "Сохраняемое переполнение"	Exception Bit Overflow Stored	15–10
Бит результата "Больше 0"	Result Bit Greater Than 0	15–5
Бит результата "Больше или равно 0"	Result Bit Greater Equal 0	15–5
Бит результата "Меньше 0"	Result Bit Less Than 0	15–5
Бит результата "Меньше или равно 0"	Result Bit Less Equal 0	15–5
Бит результата "Не равно 0"	Result Bit Not Equal 0	15–5
Бит результата "Равно 0"	Result Bit Equal 0	15–5
Включить главное управляющее реле	Master Control Relay On	16–14
Возврат	Return	16–8
Выключить главное управляющее реле	Master Control Relay Off	16–14
Выделить целую часть числа	Truncate Double Integer Part	10–16
Вызвать системную FC из окна списка	Call System FC from Box	16–4
Вызвать системный FB из окна списка	Call System FB from Box	16–4
Вызвать FB из окна списка	Call FB from Box	16–4
Вызвать FC из окна списка	Call FC from Box	16–4
Вызвать FC/SFC из катушки (без параметров)	Call FC SFC from Coil (without parameters)	16–2
Выполнить циклический сдвиг двойного слова влево	Rotate Left Double Word	12–10
Выполнить циклический сдвиг двойного слова вправо	Rotate Right Double Word	12–10
Выходная катушка	Output Coil	4–5
Вычитание вещественных чисел	Subtract Real	8–4
Вычитание двойных целых чисел	Subtract Double Integer	7–5
Вычитание целых чисел	Subtract Integer	7–4
Деактивизировать главное управляющее реле	Master Control Relay Deactivate	16–11
Деление вещественных чисел	Divide Real	8–6
Деление двойных целых чисел	Divide Double Integer	7–9
Деление целых чисел	Divide Integer	7–8
Дополнение двойного целого числа до двух	TWOs Complement Double Integer	10–13
Дополнение двойного целого числа до единицы	ONEs Complement Double Integer	10–11
Дополнение целого числа до двух	TWOs Complement Integer	10–12
Дополнение целого числа до единицы	ONEs Complement Integer	10–10
Изменить знак вещественного числа	Negate Real Number	10–14
Инвертированный бит ошибки "Недопустимая операция"	Negated Exception Bit Unordered	15–7
Инвертированный бит ошибки "Переполнение"	Negated Exception Bit Overflow	15–8
Инвертированный бит ошибки "Регистр BR"	Negated Exception Bit BR Memory	15–4

Таблица А–4. Команды KOP (Ladder Logic), упорядоченные по алфавиту в соответствии с их русскими наименованиями, с их международными соответствиями

Русское наименование	Международное наименование	№ стр.
Инвертированный бит ошибки "Сохраняемое переполнение"	Negated Exception Bit Overflow Stored	15–10
Инвертированный бит результата "Больше 0"	Negated Result Bit Greater Than 0	15–5
Инвертированный бит результата "Больше или равно 0"	Negated Result Bit Greater Equal 0	15–5
Инвертированный бит результата "Меньше 0"	Negated Result Bit Less Than 0	15–5
Инвертированный бит результата "Меньше или равно 0"	Negated Result Bit Less Equal 0	15–5
Инвертированный бит результата "Не равно 0"	Negated Result Bit Not Equal 0	15–5
Инвертированный бит результата "Равно 0"	Negated Result Bit Equal 0	15–5
Инвертировать результат логической операции (поток энергии)	Invert Power Flow	4–8
Катушка со счетчиком обратного счета	Down Counter Coil	4–14
Катушка со счетчиком прямого счета	Up Counter Coil	4–13
Катушка с таймером – формирователем задержки включения (SE)	On–Delay Timer Coil	4–17
Катушка с таймером – формирователем задержки включения с запоминанием (SS)	Retentive On–Delay Timer Coil	4–18
Катушка с таймером – формирователем задержки выключения (SA)	Off–Delay Timer Coil	4–19
Катушка с таймером – формирователем импульса (SI)	Pulse Timer Coil	4–15
Катушка с таймером – формирователем удлиненного импульса (SV)	Extended Pulse Timer Coil	4–16
Нормально замкнутый контакт	Normally Closed Contact (Address)	4–4
Нормально открытый контакт	Normally Open Contact (Address)	4–3
Обнаружение отрицательного фронта результата логической операции (RLO)	Negative RLO Edge Detection	4–21
Обнаружение положительного фронта RLO	Positive RLO Edge Detection	4–20
Обнаружение отрицательного фронта сигнала	Address Negative Edge Detection	4–23
Обнаружение положительного фронта сигнала	Address Positive Edge Detection	4–22
Округлить до ближайшего большего целого числа	Ceiling	10–17
Округлить до ближайшего меньшего целого числа	Floor	10–18
Округлить до двойного целого числа	Round to Double Integer	10–15
Открыть блок данных: DB или DI	Open Data Block: DB or DI	13–2
Перейти, если не 1	Jump–If–Not	14–5
Перейти (если 1)	Jump	14–3
Умножение вещественных чисел	Multiply Real	8–5
Умножение двойных целых чисел	Multiply Double Integer	7–7
Умножение целых чисел	Multiply Integer	7–6
Получить остаток от деления двойного целого числа	Return Fraction Double Integer	7–10
Поразрядное ИЛИ над двойными словами	(Word) Or Double Word	11–9
Поразрядное ИЛИ над словами	(Word) Or Word	11–7
Поразрядное И над двойными словами	(Word) And Double Word	11–5
Поразрядное И над словами	(Word) And Word	11–3
Поразрядное исключающее ИЛИ над двойными словами	(Word) Exclusive Or Double Word	11–13

Таблица А–4. Команды КОР (Ladder Logic), упорядоченные по алфавиту в соответствии с их русскими наименованиями, с их международными соответствиями

Русское наименование	Международное наименование	№ стр.
Поразрядное исключающее ИЛИ над словами	(Word) Exclusive Or Word	11–11
Преобразовать двоично-десятичное число в двойное целое	BCD to Double Integer	10–7
Преобразовать двоично-десятичное число в целое	BCD to Integer	10–4
Преобразовать двойное целое число в двоично-десятичное	Double Integer to BCD	10–8
Преобразовать двойное целое число в вещественное	Double Integer to Real	10–9
Преобразовать целое число в двоично-десятичное	Integer to BCD	10–5
Преобразовать целое число в двойное целое	Integer to Double Integer	10–6
Присвоить значение	Assign a Value	10–2
Промежуточный выход (коннектор)	Midline Output	4–7
Сбросить выход	Reset Coil	4–11
Сдвинуть влево двойное слово	Shift Left Double Word	12–4
Сдвинуть влево слово	Shift Left Word	12–2
Сдвинуть вправо двойное слово	Shift Right Double Word	12–4
Сдвинуть вправо двойное целое число	Shift Right Double Integer	12–9
Сдвинуть вправо слово	Shift Right Word	12–5
Сдвинуть вправо целое число	Shift Right Integer	12–7
Сложение вещественных чисел	Add Real	8–3
Сложение двойных целых чисел	Add Double Integer	7–3
Сложение целых чисел	Add Integer	7–2
Сохранить RLO в регистре BR	Save RLO to BR Memory	4–9
Сравнить вещественные числа	Compare Real (>, <, ==, <=>, <=, >=)	9–5
Сравнить двойные целые числа	Compare Double Integer (>, <, ==, <=>, <=, >=)	9–3
Сравнить целые числа	Compare Integer (>, <, ==, <=>, <=, >=)	9–2
Счетчик обратного счета	Down Counter	6–8
Счетчик прямого и обратного счета	Up–Down Counter	6–4
Счетчик прямого счета	Up Counter	6–6
Таймер S5 – формирователь задержки включения (SE)	On–Delay S5 Timer	5–10
Таймер S5 – формирователь задержки включения с запоминанием (SS)	Retentive On–Delay S5 Timer	5–13
Таймер S5 – формирователь задержки выключения (SA)	Off–Delay S5 Timer	5–15
Таймер S5 – формирователь импульса (SI)	Pulse S5 Timer	5–6
Таймер S5 – формирователь удлиненного импульса (SV)	Extended Pulse S5 Timer	5–8
Сбросить-установить триггер, RS-триггер	Reset–Set Flipflop	4–25
Установить-сбросить триггер, SR-триггер	Set–Reset Flipflop	4–24
Установить выход	Set Coil	4–10
Установить начальное значение счетчика	Set Counter Value	4–12

A.5 Список международных сокращенных наименований и сокращенных наименований SIMATIC

В таблице A–5 представлен список команд, у которых сокращенное международное имя отличается от сокращенного имени SIMATIC. В таблице эти команды перечислены в алфавитном порядке в соответствии с их полными международными именами.

Таблица A–5. Команды KOP (Ladder Logic), перечисленные в данном руководстве, с их сокращенными именами, международными и SIMATIC

Международное имя	Международное сокращение	Сокращение SIMATIC	Стр.
Down Counter	S_CD	Z_RUECK	6–8
Down Counter Coil	----(CD)	----(ZR)	4–14
Exception Bit BR Memory	BR --- ---	BIE --- ---	15–4
Extended Pulse S5 Timer	S_PEXT	S_VIMP	5–8
Extended Pulse Timer Coil	---(SE)	---(SV)	4–16
Off-Delay S5 Timer	S_OFFDT	S_AVERZ	5–15
Off-Delay Timer Coil	---(SF)	---(SA)	4–19
On-Delay S5 Timer	S_ODT	S_EVERZ	5–10
On-Delay Timer Coil	---(SD)	---(SE)	4–17
Open Data Block: DB or DI	---(OPN)	---(AUF)	13–2
Pulse S5 Timer	S_PULSE	S_IMPULS	5–6
Pulse Timer Coil	---(SP)	---(SI)	4–15
Retentive On-Delay S5 Timer	S_ODTS	S_SEVERZ	5–13
Retentive On-Delay Timer Coil	---(SS)	---(SS)	4–18
Set Counter Value	---(SC)	---(SZ)	4–12
Up Counter	S_CU	Z_VORW	6–6
Up Counter Coil	---(CU)	---(ZV)	4–13
Up-Down Counter	S_CUD	ZAehler	6–4

В Примеры программирования

Обзор главы

Раздел	Описание	Стр.
В.1	Обзор	В–2
В.2	Битовые логические операции	В–3
В.3	Таймерные команды	В–7
В.4	Операции счета и сравнения	В–11
В.5	Арифметические операции с целыми числами	В–13
В.6	Логические операции со словами	В–14

В.1 Обзор

Практические применения

Каждая из описанных в данном руководстве команд KOP (ladder logic) запускает определенную операцию. Объединяя эти команды в программу, вы можете решать широкий спектр задач автоматизации. Эта глава дает вам следующие примеры практического применения команд KOP (ladder logic):

- Управление лентой транспортера с помощью битовых логических операций
- Определение направления движения ленты транспортера с помощью битовых логических операций
- Генерирование тактовых импульсов с помощью таймерных команд
- Контроль зоны хранения с помощью операций счета и сравнения
- Решение задачи с помощью арифметических операций для целых чисел
- Установка длительности времени нагрева печи

Используемые команды

Примеры в этой главе используют следующие команды:

- Возврат --(RET)
- Выходная катушка --()
- Деление целых чисел (DIV_I)
- Катушка со счетчиком обратного счета --(CD)
- Катушка со счетчиком прямого счета --(CU)
- Катушка с таймером – формирователем удлиненного импульса --(SE)—
- Нормально замкнутый контакт --| / |--
- Нормально открытый контакт --| |--
- Обнаружение положительного фронта RLO --(P)--
- Переход, если не 1 --(JMPN)—
- Поразрядное логическое И со словами (WAND_W)
- Поразрядное логическое ИЛИ со словами (WOR_W)
- Присваивание значения (MOVE)
- Сброс выхода --(R)
- Сложение целых чисел (ADD_I)
- Сравнение целых чисел (CMP_I>=)
- Сравнение целых чисел (CMP_I<=)
- Умножение целых чисел (MUL_I)
- Установка выхода --(S)

В.2 Битовые логические операции

Управление лентой транспортера

На рисунке В -1 представлена лента транспортера, которая может приводиться в движение с помощью электродвигателя. В начале ленты находятся две кнопки: S1 – ПУСК и S2 – СТОП. В конце ленты также находятся две кнопки: S3 – ПУСК и S4 – СТОП. Лента может запускаться или останавливаться с любого конца. Кроме того, датчик S5 останавливает ленту, если предмет на ленте доходит до конца.

Символическое программирование

Вы можете составить программу управления лентой транспортера, показанной на рис. В-1, представив различные компоненты конвейерной системы с помощью символов. Если вы выберете этот метод, вы должны создать таблицу символов, чтобы увязать выбранные вами символы с абсолютными значениями (см. таблицу В–1). Символы определяются в таблице символов (см. оперативную помощь STEP 7).

Таблица В-1. Элементы символического программирования для конвейерных систем				
Компонент системы	Абсолютный адрес	Символ	Таблица символов	
Кнопка ПУСК	I 1.1	S1	I 1.1	S1
Кнопка СТОП	I 1.2	S2	I 1.2	S2
Кнопка ПУСК	I 1.3	S3	I 1.3	S3
Кнопка СТОП	I 1.4	S4	I 1.4	S4
Датчик	I 1.5	S5	I 1.5	S5
Двигатель	Q 4.0	MOTOR_ON	Q 4.0	MOTOR_ON

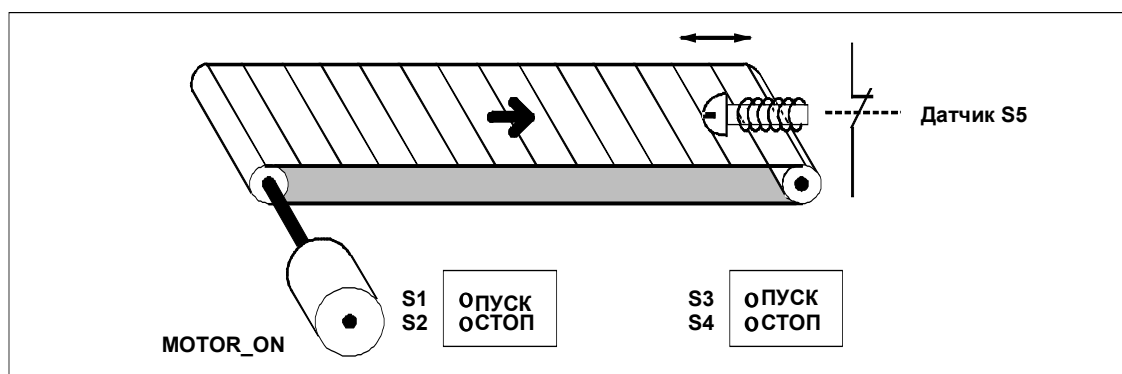


Рис. В–1. Конвейерная система

Абсолютное программирование

Вы можете написать программу для управления лентой транспортера, показанного на рис. В-1, используя абсолютные значения для представления различных компонентов конвейерной системы (см. таблицу В-2). На рис. В-2 показана программа на языке KOP (ladder logic) для управления лентой транспортера.

Таблица В-2. Элементы абсолютного программирования для конвейерной системы

Компонент системы	Абсолютный адрес
Кнопка ПУСК	I 1.1
Кнопка СТОП	I 1.2
Кнопка ПУСК	I 1.3
Кнопка СТОП	I 1.4
Датчик	I 1.5
Двигатель	Q 4.0

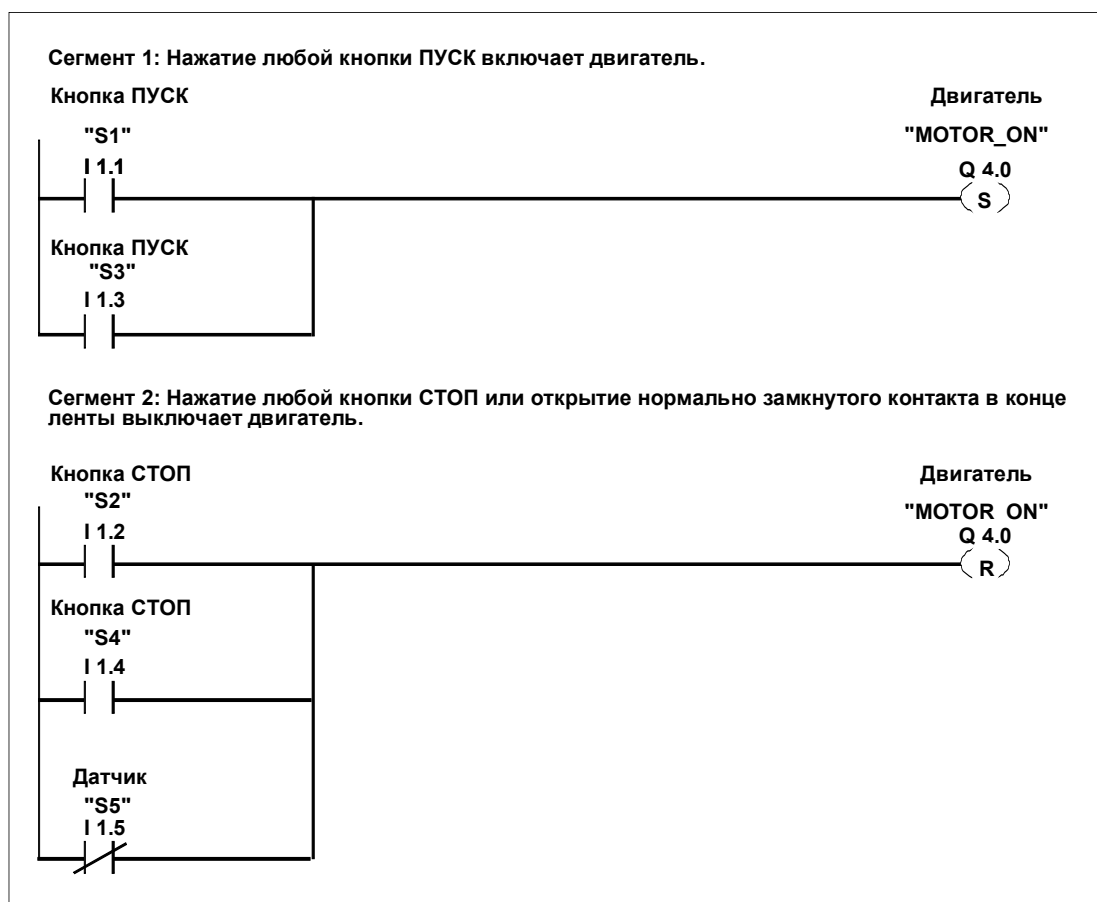


Рис. В-2. Контактный план для управления лентой транспортера

Определение направления движения ленты транспортера

На рис. В-3 показана лента транспортера, которая оснащена двумя фотоэлектрическими датчиками (РЕВ1 и РЕВ2), спроектированными для определения направления, в котором перемещается пакет на ленте. Каждый из фотодатчиков работает как нормально открытый контакт (см. раздел 4.2).

Символическое программирование

Вы можете написать программу, которая активизирует указатель направления движения конвейерной системы, изображенной на рис. В-3, используя символы, представляющие различные компоненты конвейерной системы, включая фотоэлектрические датчики для обнаружения направления. Если вы выберете этот метод, вы должны создать таблицу символов, связывающую выбранные вами символы с абсолютными значениями (см. таблицу В-3). Символы определяются в таблице символов (см. оперативную помощь STEP 7).

Таблица В-3. Элементы символического программирования для определения направления				
Компонент системы	Абсолютный адрес	Символ	Таблица символов	
Фотодатчик 1	I 0.0	РЕВ1	I 0.0	РЕВ1
Фотодатчик 2	I 0.1	РЕВ2	I 0.1	РЕВ2
Указатель движения вправо	Q 4.0	RIGHT	Q 4.0	RIGHT
Указатель движения влево	Q 4.1	LEFT	Q 4.1	LEFT
Импульсный бит памяти 1	M 0.0	PMB1	M 0.0	PMB1
Импульсный бит памяти 2	M 0.1	PMB2	M 0.1	PMB2

Абсолютное программирование

Вы можете написать программу, которая активизирует указатель направления движения конвейерной системы, изображенной на рис. В-3, используя абсолютные значения, представляющие фотодатчики для определения направления движения (см. таблицу В-4). На рис. В-4 показана программа на языке KOP (ladder logic) для управления указателем направления движения ленты транспортера.

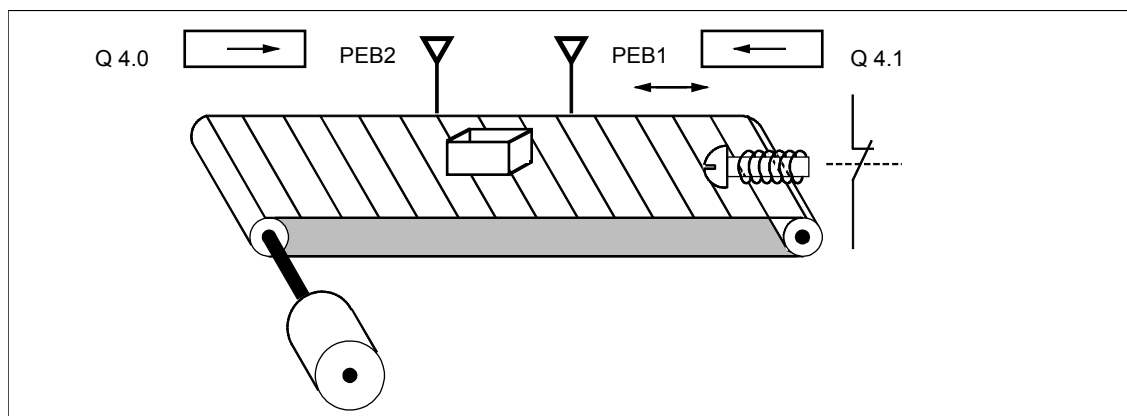
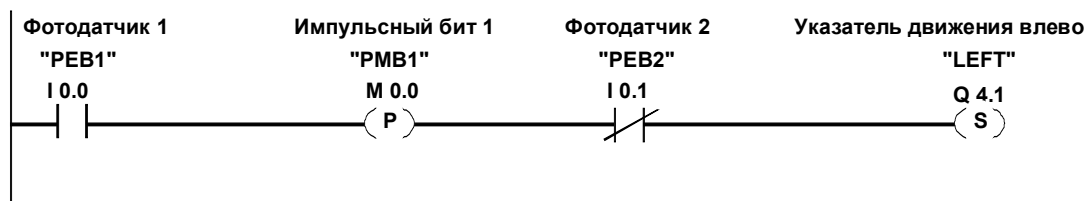


Рис. В-3. Конвейерная система с фотоэлектрическими датчиками для определения направления движения

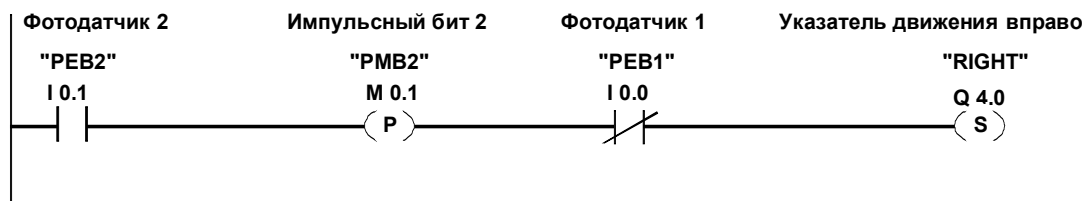
Таблица В-4. Элементы абсолютного программирования для определения направления

Компонент системы	Абсолютный адрес
Фотодатчик 1	I 0.0
Фотодатчик 2	I 0.1
Указатель движения вправо	Q 4.0
Указатель движения влево	Q 4.1
Импульсный бит памяти 1	M 0.0
Импульсный бит памяти 2	M 0.1

Сегмент 1: Если имеет место переход состояния сигнала с 0 на 1 (положительный фронт) на входе I 0.0, и при этом состояние сигнала на входе I 0.1 равно 0, то пакет на ленте движется влево.



Сегмент 2: Если имеет место переход состояния сигнала с 0 на 1 (положительный фронт) на входе I 0.1, и при этом состояние сигнала на входе I 0.0 равно 0, то пакет на ленте движется вправо. Если световой барьер одного из фотодатчиков прерывается, то пакет находится между датчиками.



Сегмент 3: Если ни один из фотобарьеров не нарушен, то пакета между датчиками нет. Указатель направления отключается.

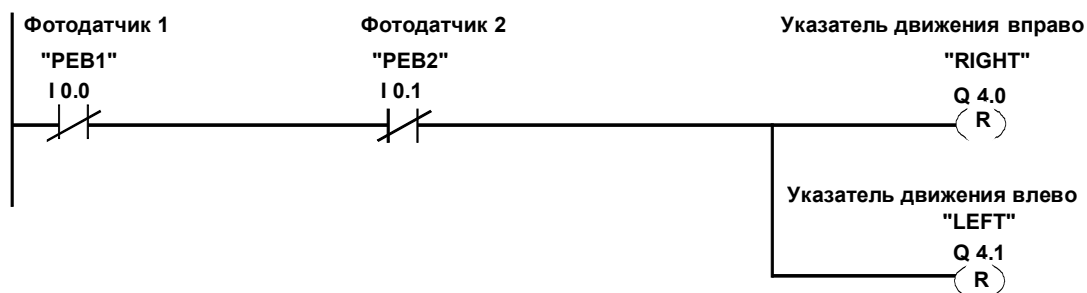


Рис. В-4. Контактный план для определения направления движения ленты транспортера

В.3 Таймерные команды

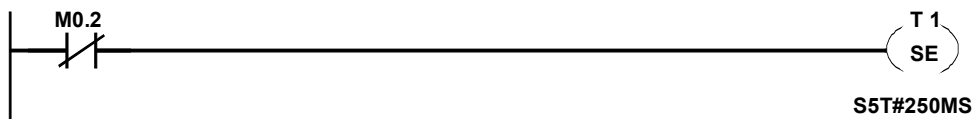
Генератор тактовых импульсов

Вы можете использовать генератор тактовых импульсов, или мигающее реле, для создания периодически повторяющегося сигнала. Генератор тактовых импульсов часто используется в системах сигнализации, управляющих миганием индикаторных ламп.

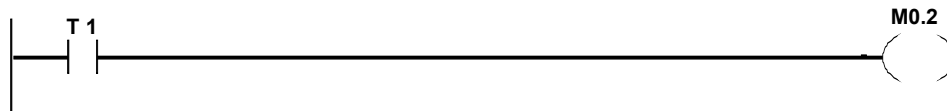
При использовании S7–300 вы можете реализовать функцию генератора тактовых импульсов, применив управляемую временем обработку в специальных организационных блоках. Пример, показанный в следующей программе на языке KOP, однако, иллюстрирует использование таймерных функций для генерирования тактовых импульсов.

Типовая программа на рис. В-5 показывает, как реализуется тактовый генератор в режиме свободных колебаний с помощью таймера (относительная длительность импульсов 1:1). Частота принимает значения, приведенные в таблице В–5.

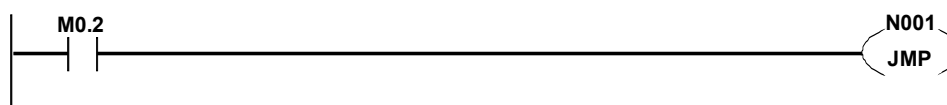
Сегмент 1: Если состояние сигнала таймера Т 1 равно 0, загрузить значение времени 250 мс в Т 1 и запустить Т 1 как формирователь продленного импульса.



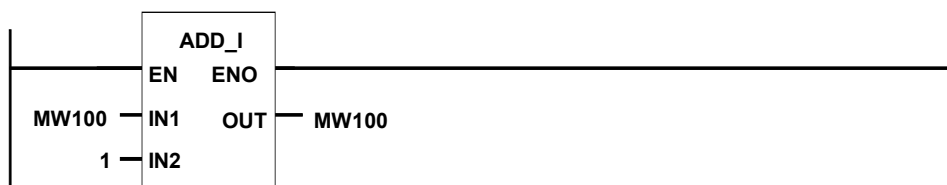
Сегмент 2: Состояние таймера временно сохраняется во вспомогательном бите памяти.



Сегмент 3: Если состояние сигнала таймера Т1 равно "1", перейти на метку N001.



Сегмент 4: Когда время таймера Т1 истекает, слово памяти 100 увеличивается на "1".



Сегмент 5: Команда MOVE позволяет вывести различные тактовые частоты на выходах от Q12.0 до Q 13.7.

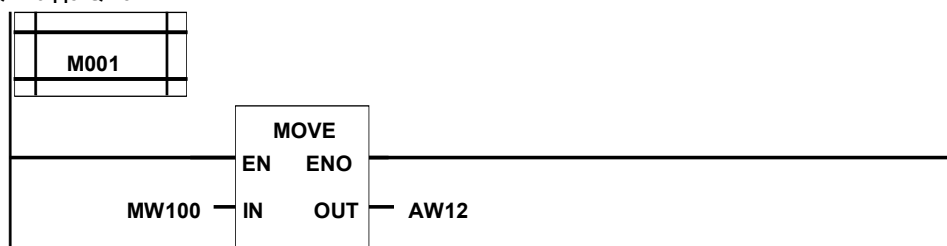


Рис. В–5. Контактный план для генерирования тактовых импульсов.

Опрос сигнала таймера Т 1 определяет результат логической операции (RLO, см. раздел 2.2), показанный на рис. В-6.

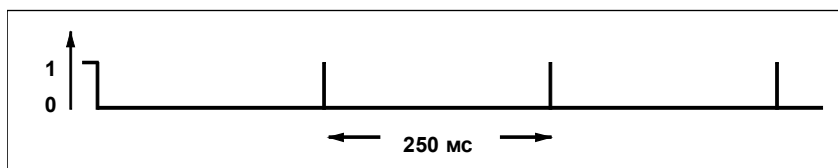


Рис. В-6. RLO для инверсного контакта Т 1 в примере генератора тактовых импульсов

Как только время таймера истекает, таймер запускается вновь. Поэтому опрос сигнала, который выполняется командой --| / |-- Т1, выдает состояние сигнала "1" очень кратковременно.

Рис. В-7 показывает, как выглядит инвертированный (обращенный) бит RLO.

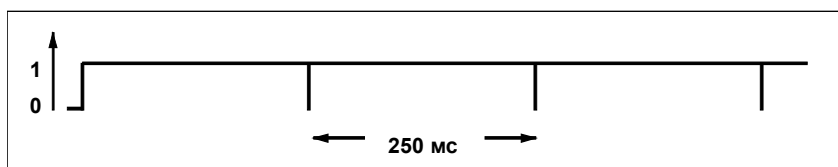


Рис. В-7. Инвертированный бит RLO таймера Т 1 в примере генератора тактовых импульсов.

Каждые 250 мс бит RLO становится равным 0. Переход игнорируется, а содержимое слова памяти MW100 увеличивается на 1.

Достижение определенной частоты

В таблице В-5 перечислены частоты, которые вы можете получить из отдельных битов байтов памяти MB101 и MB100. Сегмент 5 в контактном плане, показанном на рис. В-5, показывает, как команда MOVE позволяет увидеть различные тактовые частоты на выходах от Q12.0 до Q13.7.

Таблица В-5. Частоты для примера генератора тактовых импульсов		
Биты MB101/MB100	Частота в Гц	Длительность
M 101.0	2.0	0.5 с (250 мс вкл./250 мс выкл.)
M 101.1	1.0	1 с (0.5 с вкл./0.5 с выкл.)
M 101.2	0.5	2 с (1 с вкл./1 с выкл.)
M 101.3	0.25	4 с (2 с вкл./2 с выкл.)
M 101.4	0.125	8 с (4 с вкл./4 с выкл.)
M 101.5	0.0625	16 с (8 с вкл./8 с выкл.)
M 101.6	0.03125	32 с (16 с вкл./16 с выкл.)
M 101.7	0.015625	64 с (32 с вкл./32 с выкл.)
M 100.0	0.0078125	128 с (64 с вкл./64 с выкл.)
M 100.1	0.0039062	256 с (128 с вкл./128 с выкл.)
M 100.2	0.0019531	512 с (256 с вкл./256 с выкл.)
M 100.3	0.0009765	1024 с (512 с вкл./512 с выкл.)
M 100.4	0.0004882	2048 с (1024 с вкл./1024 с выкл.)
M 100.5	0.0002441	4096 с (2048 с вкл./2048 с выкл.)
M 100.6	0.000122	8192 с (4096 с вкл./4096 с выкл.)
M 100.7	0.000061	16384 с (8192 с вкл./8192 с выкл.)

В таблице В-6 приведен перечень состояний сигнала битов байта памяти MB101. Рис. В-8 показывает состояние сигнала бита памяти M101.1.

Таблица В-6. Состояния сигнала битов байта памяти MB101

Цикл	Состояние сигнала битов байта памяти MB101								Значение времени в мс
	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	250
1	0	0	0	0	0	0	0	1	250
2	0	0	0	0	0	0	1	0	250
3	0	0	0	0	0	0	1	1	250
4	0	0	0	0	0	1	0	0	250
5	0	0	0	0	0	1	0	1	250
6	0	0	0	0	0	1	1	0	250
7	0	0	0	0	0	1	1	1	250
8	0	0	0	0	1	0	0	0	250
9	0	0	0	0	1	0	0	1	250
10	0	0	0	0	1	0	1	0	250
11	0	0	0	0	1	0	1	1	250
12	0	0	0	0	1	1	0	0	250

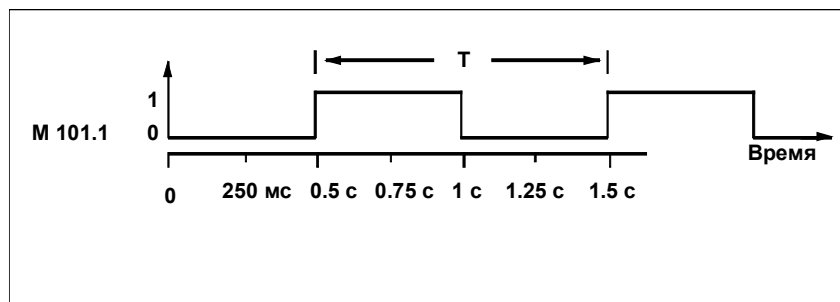


Рис. В-8. Состояние сигнала бита 1 байта MB101 (M 101.1)

В.4 Операции счета и сравнения

Зона хранения со счетчиком и компаратором

Рис. В-9 показывает систему с двумя конвейерами и зоной временного хранения между ними. Конвейер 1 транспортирует пакеты к зоне хранения. Фотодатчик в конце конвейера 1 рядом с зоной хранения определяет, сколько пакетов доставлено в зону хранения. Конвейер 2 транспортирует пакеты из зоны временного хранения к погрузочной площадке, где грузовые автомобили забирают пакеты для доставки их клиентам. Фотодатчик в конце конвейера 2 у зоны временного хранения определяет, сколько пакетов покидает зону хранения для отправки на погрузочную площадку.

Информационное табло с пятью лампочками сообщает, насколько заполнена зона временного хранения. Рис. В-10 показывает программу на языке КОР (ladder logic), которая активизирует индикаторные лампы на информационном табло.

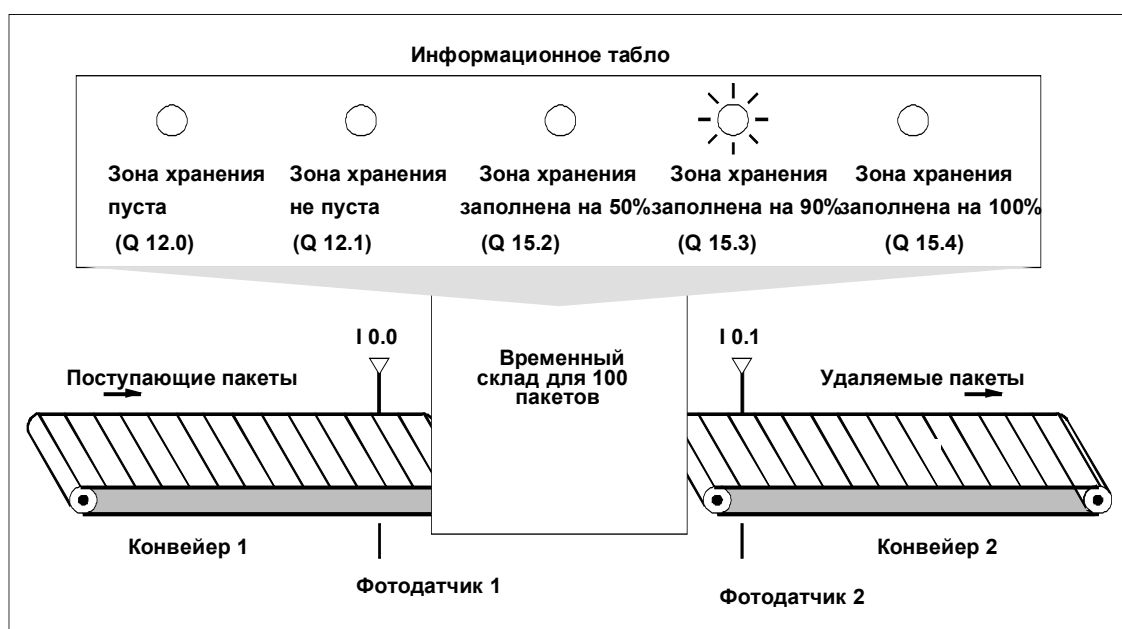
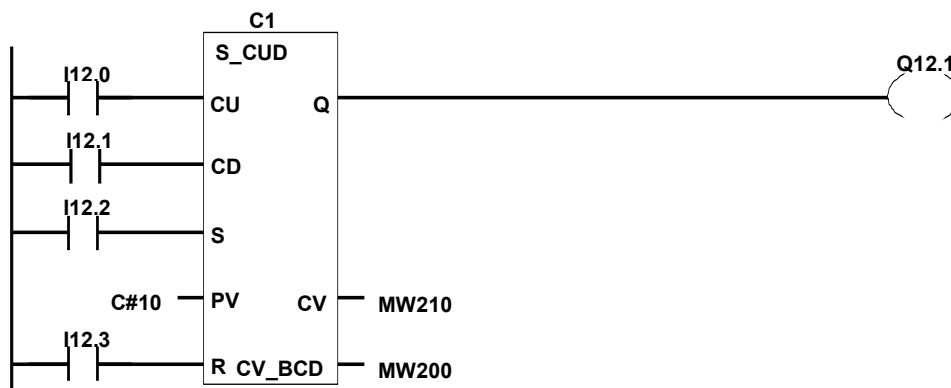
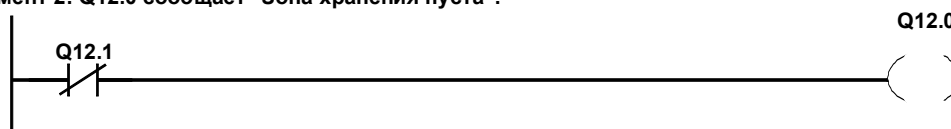


Рис. В-9. Зона хранения со счетчиком и компаратором

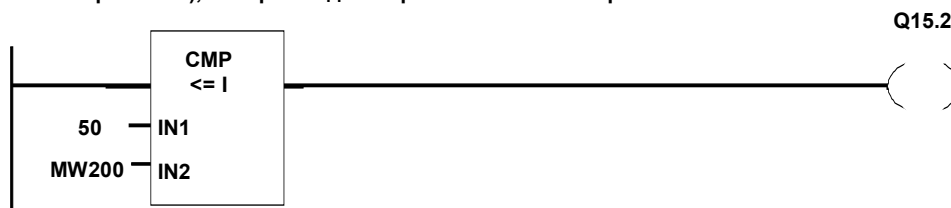
Сегмент 1: Счетчик C1 увеличивается при каждом изменении сигнала с "0" на "1" на входе CU и уменьшается при каждом изменении сигнала с "0" на "1" на входе CD. С изменением сигнала с "0" на "1" на входе S, значение счетчика устанавливается равным PV. Изменение сигнала с "0" на "1" на входе R сбрасывает значение счетчика в "0". MW200 содержит текущее значение счетчика C1. Q12.1 сообщает "Зона хранения не пуста".



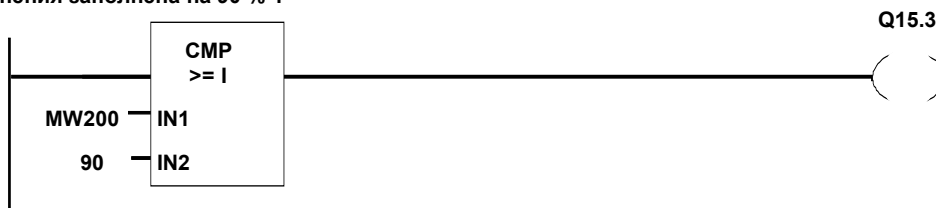
Сегмент 2: Q12.0 сообщает "Зона хранения пуста".



Сегмент 3: Если 50 меньше или равно значению счетчика (т.е. если текущее значение счетчика больше или равно 50), то горит индикаторная лампа "Зона хранения заполнена на 50 %"



Сегмент 4: Если значение счетчика больше или равно 90, то горит индикаторная лампа "Зона хранения заполнена на 90 %".



Сегмент 5: Если значение счетчика больше или равно 100, то горит индикаторная лампа "Зона хранения заполнена полностью". Используйте выход Q4.4 для блокировки конвейера 1.

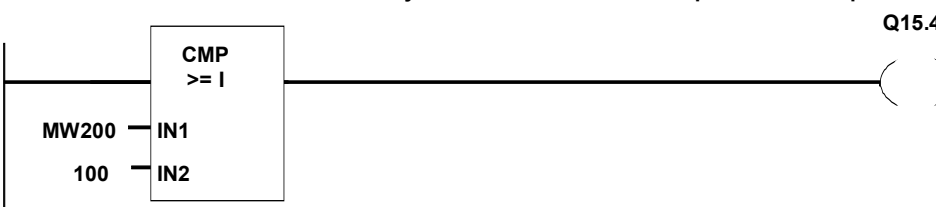


Рис. В–10. Контактный план для активизации индикаторных ламп информационной панели

В.5 Арифметические операции с целыми числами

Решение математической задачи

Пример программы, представленный на рис. В-11, показывает, как использовать три команды над целыми числами для получения того же результата, который дает следующее уравнение:

$$MD4 = \frac{(IW0 + DBW3) \times 15}{MW0}$$

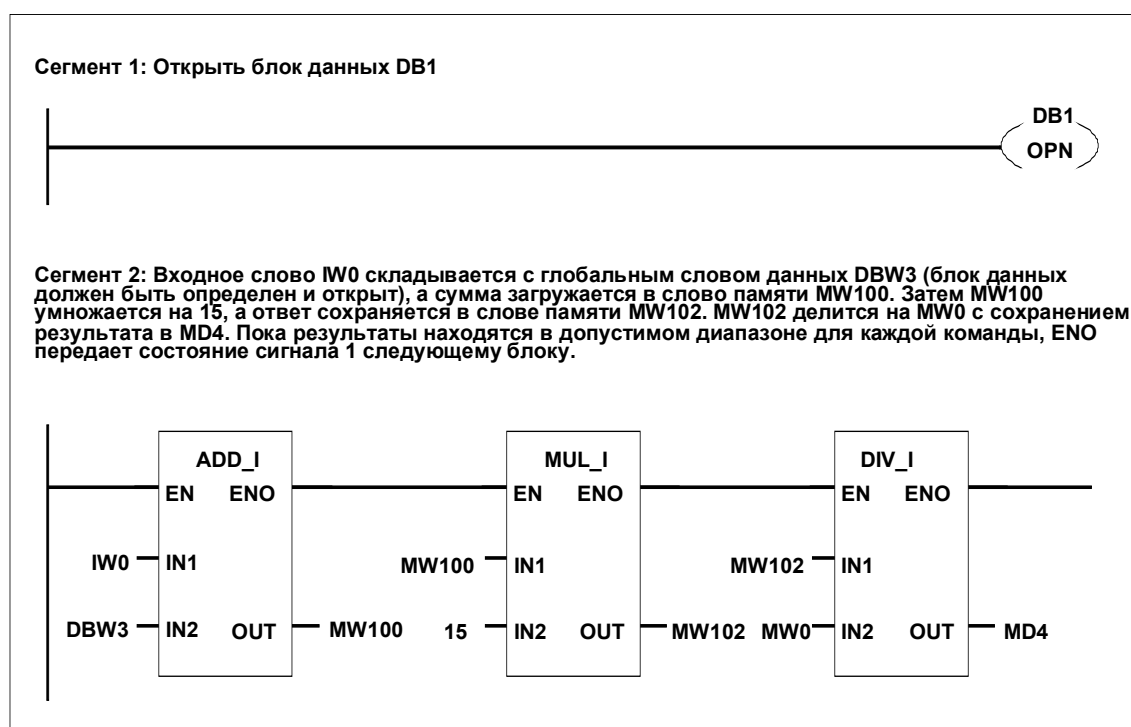


Рис. В-11. Контактный план для арифметических операций с целыми числами

В.6 Логические операции со словами

Нагревание печи

Оператор показанной на рис. В-12 печи запускает нагрев печи, нажимая кнопку ПУСК. Оператор может устанавливать длительность нагревания с помощью переключателей, показанных на рисунке. Значение, устанавливаемое оператором, указывает секунды в двоично-десятичном (BCD) формате. Таблица В-7 перечисляет компоненты системы нагрева и соответствующие им абсолютные адреса, используемые в примере программы, показанном на рис. В-13.

Таблица В-7. Компоненты системы нагрева и соответствующие абсолютные адреса	
Компонент системы	Абсолютный адрес в программе КОР
Кнопка ПУСК	I 0.7
Переключатель для единиц	от I 1.0 до I 1.3
Переключатель для десятков	от I 1.4 до I 1.7
Переключатель для сотен	от I 0.0 до I 0.3
Нагрев начат	Q 4.0

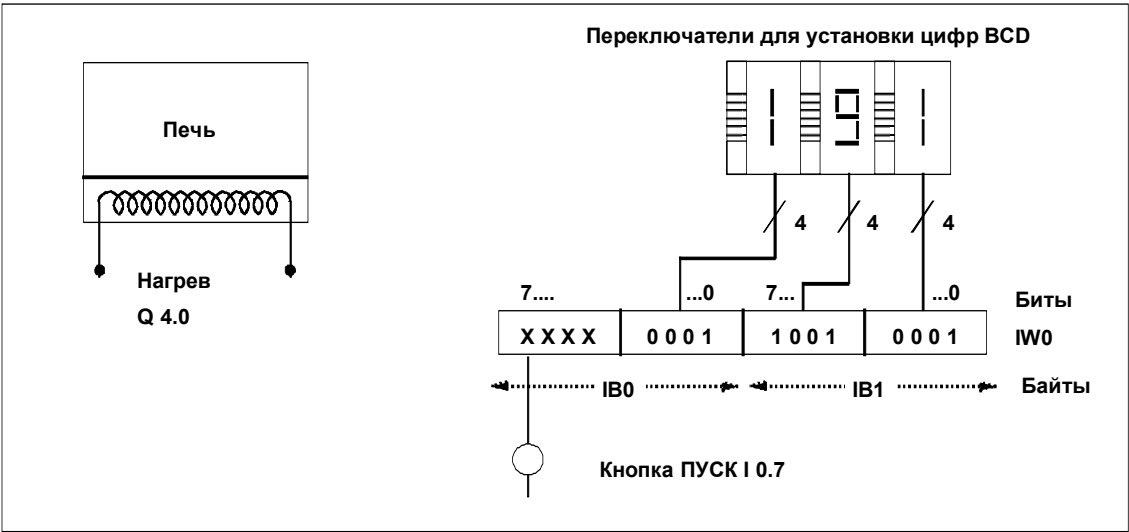
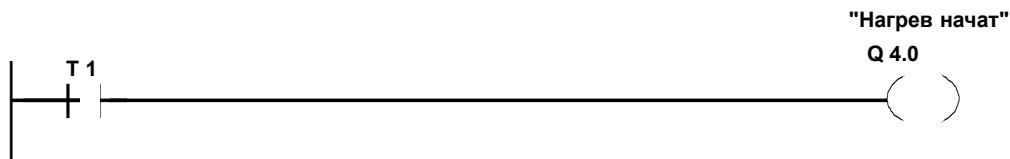
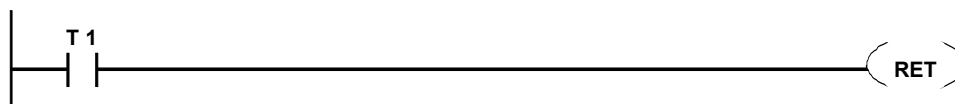


Рис. В-12. Использование входов и выходов для ограниченного временем процесса нагрева

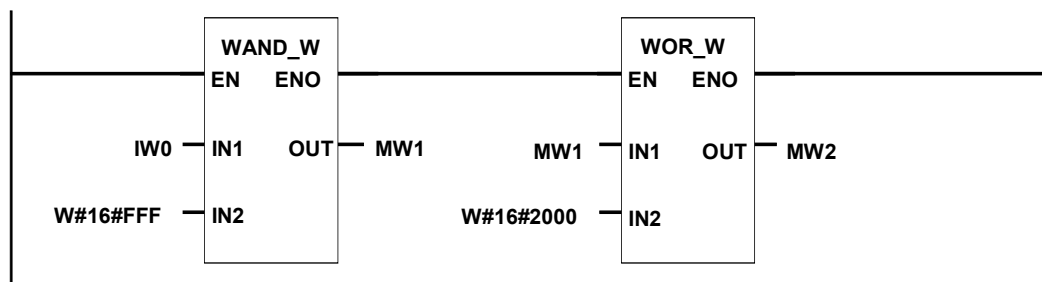
Сегмент 1: Если таймер работает, то включить нагреватель.



Сегмент 2 Если таймер работает, то команда, Return [Возврат]заканчивает обработку здесь.



Сегмент 3: Замаскировать входные биты с I 0.4 по I 0.7 (т.е. сбросить их в 0). Эти биты входов переключателей не используются. 16 битов входов переключателей комбинируются с W#16#0FFF в соответствии с командой *Поразрядное И со словами*. Результат загружается в слово памяти MW1. Для установки базы времени в секундах предустановленное значение комбинируется с W#16#2000 в соответствии с командой *Поразрядное ИЛИ со словами*, устанавливая бит 13 в 1 и сбрасывая бит 12 в 0.



Сегмент 4: Запустить таймер T 1 как формирователь удлиненного импульса, если нажата кнопка ПУСК, загружая в качестве предустановленного значения слово памяти MW2 (полученное из вышеприведенной логики).

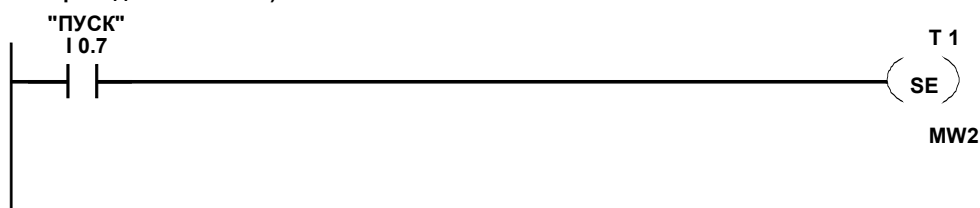


Рис. В–13. Контактный план для нагрева печи

С Литература

- /30/ Getting Started: Working with STEP 7 V5.0
[Введение: Работа со STEP 7 V5.0]
- /70/ Руководство: *S7-300 Programmable Controller*,
Hardware and Installation
[Программируемый контроллер S7-300.
Аппаратура и установка]
- /71/ Справочное руководство: *S7-300, M7-300 Programmable
Controllers*
Module Specifications
[Программируемые контроллеры S7-300, M7-300
Описания модулей]
- /72/ Instruction List: *S7-300 Programmable Controller*
[Список команд: Программируемый контроллер S7-300]
- /100/ Руководство: *S7-400/M7-400 Programmable Controllers*,
Hardware and Installation
[Программируемые контроллеры S7-400, M7-400.
Аппаратура и установка]
- /101/ Reference Manual: *S7-400/M7-400 Programmable Controllers*
Module Specifications
[Программируемые контроллеры S7-400, M7-400
Описания модулей]
- /102/ Instruction List: *S7-400 Programmable Controller*
[Список команд: Программируемый контроллер S7-400]
- /231/ Руководство: *Configuring Hardware and Communication
Connections, STEP 7 V5.0*
[Конфигурирование аппаратуры и проектирование
соединений, STEP 7 V5.0]
- /232/ Справочное руководство: *Statement List (STL) for S7-300 and
S7-400*
Programming
[Список команд (STL) для S7-300 и S7-400
Программирование]
- /234/ Руководство: *Programming with STEP 7 V5.0*
[Программирование с помощью STEP 7 V5.0]
- /235/ Справочное руководство: *System Software for S7-300 and
S7-400 System and Standard Functions*
[Системное программное обеспечение для S7-300 и S7-400
Системные и стандартные функции]
- /236/ Руководство: *FBD for S7-300 and 400*,
Programming
[FUP для S7-300 и 400, Программирование]

- /250/** Руководство: *Structured Control Language (SCL) for S7-300/S7-400, Programming*
[Язык структурного управления (SCL) для S7-300 и S7-400, Программирование]
- /251/** Руководство: *S7-GRAPH for S7-300 and S7-400, Programming Sequential Control Systems*
[S7-GRAPH для S7-300 и S7-400, Программирование систем последовательного управления]
- /252/** Руководство: *S7-HiGraph for S7-300 and S7-400, Programming State Graphs*
[S7-HiGraph для S7-300 и S7-400, Программирование графов состояния]
- /253/** Руководство: *C Programming for S7-300 and S7-400, Writing C Programs*
[Программирование на языке C для S7-300 и S7-400, Написание программ на языке C]
- /254/** Руководство: *Continuous Function Charts (CFC) for S7 and M7, Programming Continuous Function Charts*
[Схемы последовательных функций (CFC) для S7 и M7, Программирование схем последовательных функций]
- /270/** Руководство: *S7-PDIAG for S7-300 and S7-400*
«Configuring Process Diagnostics for LAD, STL, and FBD»
[S7-PDIAG для S7-300 и S7-400, "Проектирование диагностики процесса для контактного плана, списка команд и функционального плана"]
- /271/** Руководство: *NETPRO*,
«Configuring Networks»
[NETPRO,
"Проектирование сетей"]
- /800/** *DOCPRO*
Creating Wiring Diagrams (CD only)
[DOCPRO
Создание коммутационных схем (только на CD)]
- /801/** *TeleService for S7, C7 and M7*
Remote Maintenance for Automation Systems (CD only)
[TeleService для S7, C7 и M7
Дистанционное обслуживание систем автоматизации]
- /802/** PLC Simulation for S7-300 and S7-400 (CD only)
[Имитация ПЛК для S7-300 и S7-400 (только на CD)]
- /803/** Справочное руководство: *Standard Software for S7-300 and S7-400*,
STEP 7 Standard Functions, Part 2
[Стандартное программное обеспечение для S7-300 и S7-400,
Стандартные функции STEP 7, часть 2]

Глоссарий

А

Абсолютная адресация

При абсолютной адресации указывается расположение в памяти операнда, подлежащего обработке.

Адрес

Адрес (операнд) – это часть оператора STEP 7, которая указывает, с чем процессор должен выполнить команду. Адреса могут быть абсолютными или символическими.

Аккумулятор

Аккумуляторы - это регистры в CPU, которые служат в качестве промежуточной памяти для операций загрузки, передачи, сравнения, а также арифметических операций и операций преобразования.

Б

Бит переполнения

Бит состояния OV означает “переполнение”. Переполнение может возникнуть, например, после выполнения математической операции.

Бит состояния

Бит состояния (STA) хранит значение бита, к которому производится обращение. Состояние логической операции, имеющей доступ к памяти на чтение (A, AN, O, ON, X, XN) всегда равно значению бита, опрашиваемого этой операцией (бита, с которым выполняется логическая операция). Состояние логической операции, имеющей доступ к памяти на запись (S, R, =), равно значению бита, в который осуществляется запись. Если запись не имеет места, то оно равно значению бита, к которому операция обращается. Бит состояния не имеет значения для логических операций, которые не обращаются к памяти. Эти операции устанавливают бит состояния в “1” (STA = 1). Бит состояния не опрашивается командой. Он интерпретируется только во время тестирования программы (статус программы).

Бит сохраняемого переполнения

Бит состояния OS – это «бит сохраняемого переполнения слова состояния». Переполнение может иметь, например, место после выполнения математической операции.

Блок данных (DB)

Блоки данных (DB) - это области данных в программе пользователя, содержащие данные пользователя. Имеются глобальные (совместно используемые) блоки данных, к которым могут обращаться любые логические блоки, и экземплярные блоки данных, связанные с определенным вызовом функционального блока (FB). В отличие других блоков, блоки данных не содержат команд.

В

Ввод, пошаговый

При пошаговом (инкрементном) вводе блока каждая строка или элемент немедленно проверяется на наличие ошибок (например, синтаксических). Если обнаружена ошибка, то она выделяется и должна быть исправлена до завершения программирования. Пошаговый ввод возможен при программировании в виде списка команд (STL, AWL), контактного плана (LAD, KOP) и функционального плана (FBD, FUP).

Г

Глобальный блок данных (DB)

Глобальный (совместно используемый) блок данных – это DB, адрес которого при открытии загружается в адресный регистр DB. Он предоставляет память и данные для всех исполняющихся логических блоков (FC, FB и OB).

В противоположность этому, экземплярный DB проектируется для использования в качестве места хранения данных для FB, которому он поставлен в соответствие.

Д

Данные, статические

Статические данные - это локальные данные функционального блока, которые хранятся в экземплярном блоке данных и поэтому остаются незатронутыми до следующей обработки функционального блока.

Двоичный результат (BR)

Двоичный результат (binary result, BR) связывает обработку битов и слов. Это эффективный метод, позволяющий давать двоичную интерпретацию результату операции над словами и включать его в последовательность логических операций.

И

Идентификатор адреса

Идентификатор адреса – это часть адреса, содержащая различные данные, которые могут включать такие элементы, как само значение (объект данных) или размер объекта, с которым команда может, например, выполнить логическую операцию. В операторе «L IB10» IB – это идентификатор адреса («I» указывает на область входов памяти, а «B» означает один байт в этой области).

Иерархия вызовов

Прежде чем блоки могут быть обработаны, они должны быть вызваны. Последовательность и вложенность этих вызовов внутри организационного блока называется иерархией вызовов.

Исходный файл

Исходный файл (текстовый файл) – это часть программы, создаваемая с помощью графического или текстового редактора и компилируемая в исполняемую программу пользователя S7 или в машинный код для M7.

Исходный файл S7 хранится в папке «Sources [Исходные тексты]» под папкой «S7 program [Программа S7]».

К

Ключевое слово

Ключевые слова применяются при программировании с помощью исходных файлов для обозначения начала и конца блока и выделения разделов в описательной части блоков, начала комментариев к блоку и начала заголовков.

Коды условий CC 1 и CC 0

Биты CC 1 и CC 0 (коды условий) дают информацию о следующих результатах или битах:

- результат арифметической операции
- результат сравнения
- результат дискретной операции
- биты, выдвинутые операцией сдвига или циклического сдвига

Команда

Команда – это часть оператора STEP 7; она указывает, что должен делать процессор.

Контактный план (KOP, LAD)

Контактный план (Ladder Logic, LAD, KOP) – это графический язык программирования в STEP 5 и STEP 7. Его представление стандартизовано в соответствии с DIN 19239 (международный стандарт IEC 1131–1). Представление в виде контактного плана соответствует представлению в виде релейно-контактных схем. В отличие от списка команд (STL, AWL), LAD имеет более ограниченный набор команд.

Л

Логический блок

Логические блоки – это блоки в SIMATIC S7, которые содержат часть программы пользователя STEP 7. В отличие от них, блоки данных (DB) содержат только данные. Имеются следующие виды кодовых блоков: организационные блоки (OB), функциональные блоки (FB), функции (FC), системные функциональные блоки (SFB) и системные функции (SFC). Блоки хранятся в папке «Blocks [Блоки]» под папкой «S7 Program [Программа S7]».

М

Массив

Массив - это составной тип данных, состоящий из элементов данных одного типа. Эти элементы данных могут быть элементарными или составными.

Мнемоническое представление

Мнемоническое представление – это сокращенная форма отображения имен адресов и команд программирования в программе (например, «I» означает «input [вход]»). STEP 7 поддерживает международное представление (базирующееся на английском языке) и представление SIMATIC (основанное на немецких сокращениях набора команд и соглашениях об адресации, принятых в SIMATIC).

Н

Непосредственная адресация

При непосредственной адресации операнд содержит величину, с которой должна быть выполнена операция.

Пример: L27 означает загрузку в аккумулятор константы 27.

О

Область памяти

В SIMATIC S7 CPU имеет три области памяти:

- загрузочную память
- рабочую память
- системную память

Описание

Раздел описаний используется для описания локальных данных логического блока при программировании в текстовом редакторе.

П

Папка

Каталог пользовательского интерфейса Администратора SIMATIC (SIMATIC Manager), который может быть открыт и может содержать другие каталоги и объекты.

Первичный опрос

Первый опрос результата логической операции.

Программа пользователя

Программа пользователя содержит все операторы и описания и все данные для обработки сигналов, которые могут быть использованы для управления устройством или процессом. Это часть программируемого модуля (CPU, FM), которая может быть структурирована разбиением на более мелкие единицы (блоки).

Проект

Проект – это папка для всех объектов задачи автоматизации независимо от количества станций, модулей и их соединения в сети.

Прямая адресация

При прямой адресации адрес содержит расположение в памяти величины, которая должна использоваться командой.

Пример:

Ячейка памяти Q4.0 определяет бит 0 в байте 4 таблицы выходов образа процесса.

Р

Результат логической операции (RLO, VKE)

Результат логической операции (RLO, VKE) – это результат цепи логических операций, который используется для обработки других двоичных сигналов. Исполнение определенных команд целиком зависит от предшествующего RLO.

С

Сегмент

Сегменты подразделяют блоки LAD и FBD на законченные цепи тока, а блоки списка команд (STL) на удобочитаемые единицы.

Символ

Символ - это имя, которое может быть определено пользователем при соблюдении определенных синтаксических правил. После определения (например, в качестве переменной, типа данных, метки перехода, блока) оно может применяться для программирования и в для функций взаимодействия с оператором.

Пример: адрес: I 5.0, тип данных: BOOL, символ: Кнопка аварийного отключения.

Символическая адресация

При символической адресации адрес, подлежащий обработке, обозначается символом (в противоположность абсолютному адресу).

Системная функция (SFC)

Системная функция – это функция (без памяти), которая встроена в операционную систему CPU S7 и, если необходимо, может быть вызвана из программы пользователя STEP 7 подобно функции (FC).

Системный функциональный блок (SFB)

Системный функциональный блок (SFB) – это функция (с памятью), которая встроена в операционную систему S7 и, если необходимо, может быть вызвана из программы пользователя STEP 7 подобно функциональному блоку (FB).

Слово состояния

Слово состояния – это часть регистра CPU. Оно содержит информацию о состоянии и ошибках, которая отображается при выполнении определенных команд STEP 7. Биты состояния могут считываться и записываться пользователем, биты ошибок можно только считывать.

Список команд (STL, AWL)

Список команд (STL, AWL) – это текстовое представление языка программирования STEP 7, подобное машинному коду. STL – это язык ассемблера STEP 5 и STEP 7. Если вы программируете на STL, то отдельные операторы представляют фактические шаги, которыми CPU исполняет программу.

Справочные данные

Справочные данные служат для контроля программы CPU и включают в себя список перекрестных ссылок, списки занятости, структуру программы, список не использованных адресов и список адресов без символов.

Станция

Станция – это устройство, которое может быть подключено к одной или нескольким подсетям; например, программируемый контроллер, устройство программирования или станция оператора.

Структура программы пользователя

Структура программы пользователя описывает иерархию вызовов блоков внутри программы S7 и обеспечивает обзор применяемых блоков и их зависимостей.

Т

Таблица описания переменных

В таблице описания переменных описываются локальные данные логического, блока, если программирование производится в редакторе пошагового ввода.

Таблица переменных (VAT)

В таблице переменных собраны переменные, которые вы хотите наблюдать и изменять, устанавливая соответствующие форматы.

Таблица символов

Таблица для сопоставления символов адресам глобальных данных и блоков. Примеры: Аварийное отключение (символ) – I 1.7 (адрес) или Регулятор (символ) – SFB24 (блок).

Тип данных

Тип данных определяет, как значение переменной или константы должно применяться в программе пользователя.

В SIMATIC STEP 7 пользователю предоставляются в распоряжение два вида типов данных (IEC 1131-3):

- элементарные типы данных

- составные типы данных

Тип данных, составной

Составные типы данных создаются пользователем с помощью описания типа. Они не имеют собственного имени и поэтому не могут применяться повторно. Они могут быть массивами или структурами. Сюда же относятся типы данных STRING и DATE_AND_TIME.

Тип данных, элементарный

В соответствии с IEC 1131-3 элементарные типы данных - это predetermined типы данных.

Примеры:

- тип данных "BOOL" определяет двоичную переменную ("бит")
- тип данных "INT" определяет 16-битовую переменную с фиксированной точкой.

Типы данных пользователя (UDT)

Типы данных пользователя – это специальные структуры данных, которые вы можете создавать сами и, после их определения, использовать их во всей программе. Они могут использоваться как элементарные или составные типы данных а описании переменных логических блоков (FC, FB, OB) или как шаблоны для создания блоков данных с такой же структурой данных.

У

Указатель

С помощью указателя вы можете распознать адрес переменной. Указатель содержит идентификатор вместо значения. Если вы параметрическому типу Pointer [Указатель] ставите в соответствие фактический параметр, то вы передаете адрес в памяти. С помощью STEP 7 Вы можете ввести указатель в формате указателя или просто как идентификатор (например, M50.0). В следующем примере показан формат указателя, с помощью которого обращаются к данным, начиная с M 50.0:

R#M50.0

Ф

Фактический параметр

Фактические параметры заменяют формальные параметры при вызове функциональных блоков (FB) или функций (FC).

Пример: Формальный параметр «Start» заменяется фактическим параметром «I3.6».

Формальный параметр

Формальный параметр - это метка-заполнитель для фактического параметра в логических блоках. В функциональных блоках (FB) и функциях (FC) формальные параметры описываются пользователем, а в системных функциональных блоках (SFB) и системных функциях (SFC) они уже имеются. При вызове блока формальным параметрам ставятся в соответствие фактические параметры, так что вызываемый блок работает с этим фактическими значениями. Формальные параметры относятся к локальным данным блока и делятся на входные, выходные и проходные (in/out) параметры.

Функциональный блок (FB)

В соответствии со стандартом Международной электротехнической комиссии IEC 1131-3, функциональные блоки – это логические блоки с "памятью" (т.е. у них есть статические данные). Функциональный блок позволяет передавать параметры в программе пользователя, т.е. он пригоден для программирования часто повторяющихся сложных функций, например, регулирования по замкнутому контуру или выбора режима работы. Так как у функционального блока есть память (экземплярный блок данных), то вы можете обратиться к его параметрам (например, к выходам) в любое время и в любой точке программы пользователя.

Функциональный план (FBD, FUP)

Функциональный план (Function Block Diagram, FBD, FUP) – это один из языков программирования в STEP 5 и STEP 7. FBD представляет логику в виде блоков, известных из булевой алгебры. Кроме того, сложные функции (например, математические) могут быть представлены в непосредственном соединении с логическими блоками. Программы, созданные с помощью FBD, могут быть также преобразованы в представления на других языках программирования (например, в виде контактного плана).

Функция (FC)

В соответствии со стандартом Международной электротехнической комиссии IEC 1131-3, функции – это логические блоки без "памяти" (т.е. у них нет статических данных). Функции позволяют передавать параметры в программе пользователя, т.е. они пригодны для программирования часто повторяющихся сложных функций, например, расчетов. Важно: Так как у функции нет памяти, то вы должны продолжить обработку вычисленных значений непосредственно после вызова функции.

Ц

Цепь логических операций

Цепь логических операций – это часть программы пользователя, которая начинается битом FC, имеющим сигнальное состояние 0, и которая заканчивается, когда команда или событие сбрасывает бит FC в 0. Когда CPU выполняет первую команду в цепи логических операций, бит FC устанавливается в 1. Некоторые команды, например, команды вывода (установка, сброс, присваивание) сбрасывают бит FC в 0. См. Первичный опрос.

Цепь тока

Характерная особенность языка программирования *Контактный план* (KOP, Ladder Logic). Цепи тока содержат контакты и катушки. В цепь могут быть вставлены также сложные элементы (напр., математические функции) в форме “блоков”. Цепи тока подключены к токовым шинам.

Э

Экземпляр

“Экземпляром” называется вызов функционального блока. Каждому вызову ставится в соответствие экземплярный блок данных.

Экземплярный блок данных (DB)

Экземплярный блок данных хранит формальные параметры и статические данные функциональных блоков. Экземпляр блока данных может быть поставлен в соответствие одному вызову функционального блока или иерархии вызовов функциональных блоков.

М

Master Control Relay

Master Control Relay (MCR) – главное управляющее реле – применяется в релейных контактных планах для активизации и деактивизации потока сигнала (цепи тока). Деактивированная цепь тока соответствует последовательности команд, которая записывает нулевое значение вместо рассчитанного, или последовательности команд, которая оставляет неизменным существующее значение памяти.

С

S7 Program (Программа S7)

Папка для блоков, исходных файлов и схем для программируемых контроллеров S7. Программа S7 включает в себя также таблицу символов.

