SIEMENS

SIMATIC

Контактный план (КОР) для S7-300 и S7-400 Программирование

Справочное руководство

Это руководство является частью документации с заказным номером:
6ES7810-4CA07-8BW1

Редакция 01/2004
A5E00261407-01

Указания по технике безопасности

Данное руководство содержит указания, которые вы должны соблюдать для обеспечения собственной безопасности, а также защиты от повреждений продукта и связанного с ним оборудования. Эти замечания выделены предупреждающим треугольником и представлены, в соответствии с уровнем опасности следующим образом:

Опасность
указывает, что если не будут приняты надлежащие меры предосторожности, то это **приведет** к гибели людей, тяжким телесным повреждениям или существенному имущественному ущербу.

Предупреждение
указывает, что при отсутствии надлежащих мер предосторожности это **может привести** к гибели людей, тяжким телесным повреждениям или к существенному имущественному ущербу.

Осторожно
указывает, что возможны легкие телесные повреждения и нанесение небольшого имущественного ущерба при непринятии надлежащих мер предосторожности.

Осторожно
указывает, что возможно повреждение имущества, если не будут приняты надлежащие меры безопасности.

Замечание
привлекает ваше внимание к особо важной информации о продукте, обращении с ним или к соответствующей части документации.

Квалифицированный персонал

К монтажу и работе на этом оборудовании должен допускаться только **квалифицированный персонал**. Квалифицированный персонал – это люди, которые имеют право вводить в действие, заземлять и маркировать электрические цепи, оборудование и системы в соответствии со стандартами техники безопасности.

Надлежащее использование

Примите во внимание следующее:

Предупреждение
Это устройство и его компоненты могут использоваться только для целей, описанных в каталоге или технической документации, и в соединении только с теми устройствами или компонентами других производителей, которые были одобрены или рекомендованы фирмой Siemens.
Этот продукт может правильно и надежно функционировать только в том случае, если он правильно транспортируется, хранится, устанавливается и монтируется, а также эксплуатируется и обслуживается в соответствии с рекомендациями.

Товарные знаки

SIMATIC®, SIMATIC HMI® и SIMATIC NET® - это зарегистрированные товарные знаки SIEMENS AG.

Некоторые другие обозначения, использованные в этих документах, также являются зарегистрированными товарными знаками; права собственности могут быть нарушены, если они используются третьей стороной для своих собственных целей.

Siemens AG
Департамент автоматизации и приводов
Промышленные системы автоматизации
П\я 4848, D- 90327, Нюрнберг

Отказ от ответственности

Мы проверили содержание этого руководства на соответствие с описанным аппаратным и программным обеспечением. Так как отклонения не могут быть полностью исключены, то мы не можем гарантировать полного соответствия. Однако данные, приведенные в этом руководстве, регулярно пересматриваются, и все необходимые исправления вносятся в последующие издания. Мы будем благодарны за предложения по улучшению содержания.

Siemens Aktiengesellschaft
A5E00261409-01

Excellence in Automation & Drives: Siemens

Предисловие

Цель руководства

Это руководство поможет Вам при разработке прикладных программ на языке программирования KOP (контактный план).

Кроме того, в этом руководстве имеется справочный раздел по элементам языка программирования KOP, в котором описываются синтаксис и принцип работы отдельных инструкций языка.

Требования к начальной подготовке

Это руководство рассчитано на программистов - разработчиков S7-программ, пусконаладчиков и обслуживающий персонал.

Для понимания излагаемого материала желательно наличие общих знаний в области техники автоматизации.

Дополнительно желательно иметь опыт работы с компьютером и знание другого подобного PC оборудования (например программаторов) с операционными системами MS Windows 2000 Professional или MS Windows XP Professional.

Область примененения руководства

Это руководство разработано для программного пакета STEP 7 версии 5.3.

Соответствие стандартам

KOP соответствует языку ”“Kontakt Plan” ” (“Контактный план”), определенному в стандарте Международной электротехнической комиссии IEC 1131–3. По поводу детальных подробностей обратитесь к таблице стандартов, находящейся в файле NORM_TBL.WRI пакета STEP 7.

Требования

Для эффективного использования этого руководства, Вы должны быть знакомы с теорией программирования на S7 , описанной во встроенной помощи STEP 7. Языковые пакеты также используют программное обеспечение STEP 7, которое Вы должны изучить по соответствующей документации.

Это руководство является частью пакета документации "STEP 7 Reference". В следующей таблице приведен обзор STEP 7 документации:

Документация	**Содержание**	**Заказной номер**
Базовая информация по STEP 7: • Работа со STEP 7 V5.3, Первые шаги • Программирование в STEP 7 V5.3 • Конфигурация аппаратной части и коммуникаций в STEP 7 V5.3 • От S5 к S7, Руководство по конвертации	Базовая информация для технического персонала , описывающая методы выполнения задач управления со STEP 7 на программируемых контроллерах S7-300/400.	6ES7810-4CA07-8BW0
STEP 7 Справочники: • Руководства по Контактному плану (KOP)/Функциональному плану (FBD)/Списку инструкций (STL) для S7-300/400 • Стандартные и системные функции S7-300/400	Справочная информация по описанию языков программирования LAD, FBD и STL, а также стандартных и системных функций STEP 7.	6ES7810-4CA07-8BW1

Встроенная справка	**Содержание**	**Заказной номер**
Help on STEP 7	Базовая информация по программированию и конфигурированию аппаратной части STEP 7 в форме встроенной справки.	Часть стандартного программного обеспечения STEP 7.
Справочная информация по STL/KOP/FBD Справочная информация по SFBs/SFCs Справочная информация по организационным блокам.	Контекстная справочная информация.	Часть программного обеспечения STEP 7.

Online Помощь

Руководство соответствует встроенной в стандартное программное обеспечение помощи. Эта встроенная помощь предоставляет Вам детальную информацию по использованию программного обеспечения.

Встроенная система помощи доступна пользователю следующими способами:

- Контекстная помощь предоставляет помощь в контексте с текущим объектом, например, по открытому диалоговому окну . Вы можете открыть контекстную помощь через команду меню **Help > Context-Sensitive Help**, нажатием клавиши F1 или с помощью знака вопроса из панели инструментов.
- Вы можете вызвать помощь по STEP 7 с использованием команды меню **Help > Contents** или кнопки "Help on STEP 7" окна контекстно-зависимой помощи.
- Вы можете открыть словарь терминов приложений STEP 7 с помощью кнопки "Glossary".

Это руководство является выборкой из "Help on Ladder Logic ". Руководство, как и встроенная помощь имеют одинаковую структуру, поэтому, можно легко перейти от руководства к встроенной помощи.

Дальнейшая поддержка

Если у Вас возникают технические вопросы, пожалуйста свяжитесь с Вашим представительством Siemens .

Вы можете найти контактное лицо через:

http://www.siemens.com/automation/partner

Учебные центры

Siemens предлагает широкий спектр учебных курсов по изучению систем автоматизации SIMATIC S7. Пожалуйста свяжитесь с вашим региональным учебным центром или нашим центральным учебным центром в D 90327 Нюрнберге, Германия: Москва,Россия

Телефон: +49 (911) 895-3200. (095) 737-23-88

Internet: http://www.sitrain.com http://www.aud.ru

A&D Техническая поддержка

Всемирная, круглосуточная:

Всемирная (Nuernberg) **техническая поддержка** 24 часа в день, 365 дней в году Телефон: +49 (180) 5050-222 Факс: +49 (180) 5050-223 E-Mail: adsupport@ siemens.com GMT: +1:00		
Европа/Африка (Nuernberg) **техническая поддержка** Местное время:Пн.-Пт. 8:00 - 17:00 Телефон: +49 (180) 5050-222 Fax: +49 (180) 5050-223 E-Mail: adsupport@ siemens.com GMT: +1:00	**United States** (Johnson City) **техническая поддержка** Местное время:Пн.-Пт. 8:00 - 17:00 Телефон: +1 (423) 262 2522 Fax: +1 (423) 262 2289 E-Mail: simatic.hotline@ sea.siemens.com GMT: -5:00	**Asia / Australia** (Beijing) **техническая поддержка** Местное время:Пн.-Пт. 8:00 - 17:00 Телефон: +86 10 64 75 75 75 Fax: +86 10 64 74 74 74 E-Mail: adsupport.asia@ siemens.com GMT: +8:00
Языки , используемые на SIMATIC Hotlines, английский или немецкий.		

Сервис и поддержка в Интернете

В дополнение к нашей документации, мы предлагаем наши Know-how online в Интернете на сайте:

http://www.siemens.com/automation/service&support

где Вы найдете следующее:

- Бюллетень, предоставляющий вам обновляемую информацию по Вашим продуктам.
- Необходимые документы с помощью функции Search в Service & Support.
- Форум, где пользователи и специалисты со всего мира обмениваются опытом.
- Ваши местные представительства департамента автоматизации и приводов.
- Информацию по сервису, ремонту, запчастям и прочему в разделе "Services".

Содержание

1 Битовые логические инструкции

1.1 Обзор битовых логических инструкций

Описание

Битовые логические инструкции работают с двумя числами - 1 и 0. Эти две цифры образуют базис системы счисления, называемой двоичной системой. Цифры 1 и 0 называются двоичными цифрами (**bi**nary digi**ts**) или просто битами. При работе со схемами, использующими контакты и катушки, значение 1 означает активное состояние или протекание тока, а 0 – неактивное состояние или отсутствие протекания тока.

Битовые логические инструкции интерпретируют состояния сигналов 1 и 0 и комбинируют их по правилам булевой логики. Эти комбинации дают результат 1 или 0, называемый Результатом Логической Операции (RLO). Битовые логические инструкции предоставляют в распоряжение следующие функции:

---| |--- Нормально открытый контакт

- ---| / |--- Нормально замкнутый контакт
- ---(SAVE) Сохранение RLO в бите BR
- XOR ИСКЛЮЧАЮЩЕЕ ИЛИ
- ---( ) Выходная катушка
- ---(#)--- Промежуточный выход (коннектор)
- ---|NOT|--- Инверсия результата логической операции

Выполнение следующих инструкций производится только при RLO = 1:

- ---(S) Установить выход в 1
- ---(R) Сбросить выход в 0
- SR S/R Триггер
- RS R/S Триггер

Другие инструкции работают при нарастающем или падающем фронте:

---(N)--- Выделение отрицательного фронта RLO

---(P)--- Выделение положительного фронта RLO

- NEG Выделение отрицательного фронта сигнала
- POS Выделение положительного фронта сигнала
- Промежуточное чтение
- Промежуточная запись

1.2 ---| |--- Нормально открытый контакт (Адрес)

Обозначение

<адрес>

---| |---

Параметр	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, L, D, T, C	Адрес опрашиваемого бита

Описание

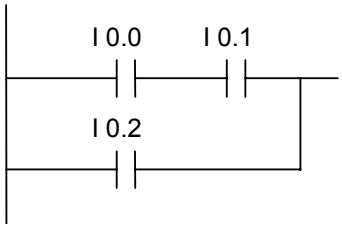
---| |--- :(Нормально открытый контакт) будет замыкаться при состоянии бита, указанного в качестве **<адреса>**, равном 1 . Если состояние сигнала по указанному адресу равно 1, то контакт замкнут, и результат логической операции (RLO)равен 1. Если состояние сигнала по указанному адресу равно 0, то контакт разомкнут, и команда дает результат логической операции (RLO) равный 0.

При использовании команды **---| |---** в последовательной цепи, результат опроса сопрягается с битом RLO по логическому И. При использовании команды **---| |---** в параллельной цепи, результат опроса сопрягается с битом RLO по логическому ИЛИ.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	X	X	X	1

Пример



Ток протекает, если выполняется одно из следующих условий:

Состояние сигнала на входах I0.0 И I0.1 равно 1

ИЛИ равно 1 состояние сигнала на входе I0.2

1.3 ---| / |--- Нормально замкнутый контакт (Адрес)

Обозначение

<адрес>

---| / |---

Параметр	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, L, D, T, C	Адрес опрашиваемого бита

Описание

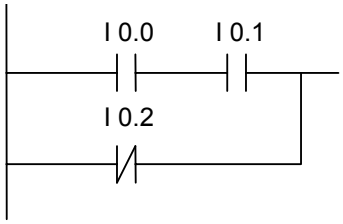
---| / |--- (Нормально замкнутый контакт) будет замыкать цепь при состоянии бита, указанного в качестве **<адреса>**, равном 0 . Если состояние сигнала по указанному адресу равно 0, то контакт замкнут, и результат логической операции (RLO)равен 1. Если состояние сигнала по указанному адресу равно 1, то контакт разомкнут, и команда дает результат логической операции (RLO) равный 0.

При использовании команды **---| / |---** в последовательной цепи, результат опроса сопрягается с битом RLO по логическому И. При использовании команды **---| / |---** в параллельной цепи, результат опроса сопрягается с битом RLO по логическому ИЛИ.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	X	X	X	1

Пример



Ток протекает, если выполняется одно из следующих условий:

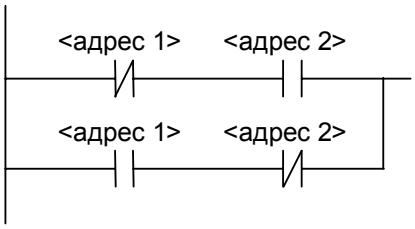
Состояние сигнала на входах I0.0 И I0.1 равно 1

ИЛИ равно 0 состояние сигнала на входе I0.2

1.4 XOR : Логическая инструкция исключающее ИЛИ

Для логической функции “Исключающее ИЛИ” (XOR) должна быть составлена следующая комбинация нормально открытых и нормально закрытых контактов:

Обозначение

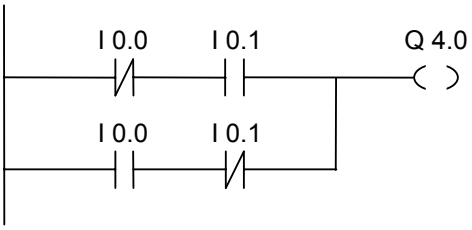


Параметр	Тип данных	Область памяти	Описание
<адрес1>	BOOL	I, Q, M, L, D, T, C	Адрес опрашиваемого бита
<адрес2>	BOOL	I, Q, M, L, D, T, C	Адрес опрашиваемого бита

Описание

XOR :(Битовая инструкция “**Исключающее ИЛИ**”) формирует RLO равным "1" при различном состоянии опрашиваемых сигналов.

Пример



Выход Q4.0 равен "1" если I0.0 = "0" И I0.1 = "1" ИЛИ I0.0 = "1" И I0.1 = "0".

1.5 --|NOT|-- Инверсия результата логической операции

Обозначение

---|NOT|---

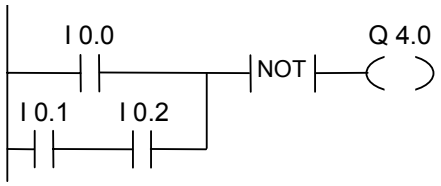
Описание

---|NOT|--- Инструкция инверсия результата логической операции выполняет изменение на противоположное значение результата логической операции RLO.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	-	1	X	-

Пример



Выход Q4.0 равен "0" если выполнено одно из следующих условий:

Состояние сигнала на входе I0.0 равно 1

ИЛИ состояние сигнала равно 1 на входе I0.1 И на входе I0.2.

1.6 ---( ) Выходная катушка

Обозначение

<адрес>

---( )

Параметр	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, L, D	Управляемый бит

Описание

---( ) :(**Выходная катушка**) работает как катушка в цепи управления релейно-контактной схемы. Если к катушке подводится ток (RLO = 1), бит **<адрес>** устанавливается в "1". Если к катушке не подводится ток (RLO = 0), бит **<адрес>** устанавливается в "0". Выходную катушку можно установить только на правом конце логической цепи. Возможно использование нескольких выходных катушек (максимум 16) . Вы можете инвертировать выход с помощью инструкции---|NOT|--- .

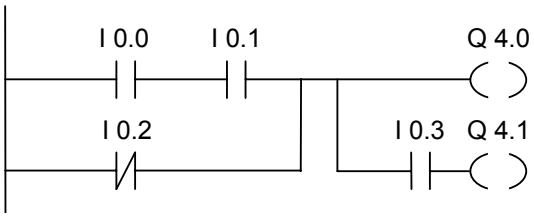
Зависимость от главного управляющего реле MCR (Master Control Relay)

Эта зависимость имеет место только при размещении выходной катушки внутри активированной зоны главного управляющего реле. Внутри зоны действия MCR, если функция MCR активирована и ток протекает на выходную катушку , бит адреса устанавливается на подводимое к нему значение RLO. Если функция MCR деактивирована, логический "0" записывается в выходной адрес независимо от подводимого к нему RLO.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	0	X	-	0

Пример



Сигнал на выходе Q4.0 равен "1" при выполнении одного из условий:

Состояние сигнала "1" на входах I0.0 И I0.1

ИЛИ состояние сигнала "0" на входе I0.2.

Сигнал на выходе Q4.1 равен "1" при выполнении одного из условий:

Состояние сигнала "1" на входах I0.0 И I0.1 И сигнал "1" на входе I0.3

ИЛИ состояние сигнала "0" на входе I0.2 И сигнал "1" на входе I0.3

При работе внутри MCR зоны:

Если функция MCR включена, выходы Q4.0 и Q4.1 устанавливаются в соответствии с вышеописанными условиями.

При выключенной MCR (=0), выходы Q4.0 и Q4.1 сбрасываются в 0 независимо от протекания тока.

1.7 ---(#)--- Коннектор

Обозначение

<адрес>

---(#)---

Параметр	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, *L, D	Управляемый бит

* Вы можете использовать временные переменные, только если они описаны в таблице декларации блоков (FC, FB, OB) в разделе TEMP.

Описание

---(#)--- (**Коннектор**) – это промежуточный присваивающий элемент, сохраняющий RLO в указанном битовом **<адресе>**. Этот промежуточный элемент запоминает результат логической операции предыдущей логической цепочки. При последовательном соединении с другими контактами ---(#)--- работает как обычный контакт.

Элемент ---(#)--- никогда не может ставиться один перед катушкой, в конце сегмента или в конце открытой ветви. Инверсный ---(#)--- можно получить с помощью элемента ---|NOT|--- (инверсия RLO).

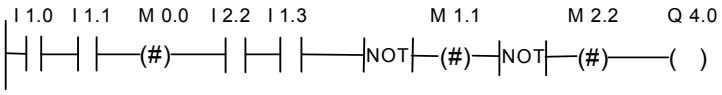
Зависимость от MCR

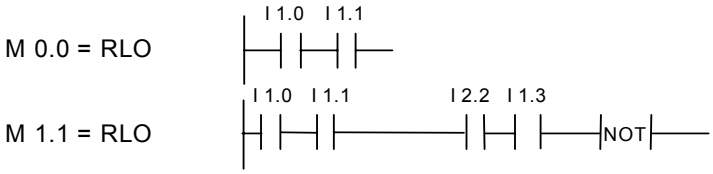
Функция MCR влияет только при расположении коннектора внутри MCR зоны. Внутри активированной зоны MCR , при ее включении, адрес принимает значение подводимого RLO. Внутри деактивированной зоны MCR логический "0" записывается в указанный адрес независимо от подводимого RLO.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	0	X	-	1

Пример





M 2.2 сохраняет RLO всей логической цепочки

1.8 ---(R) Сброс бита

Обозначение

<адрес>

---(R)

Параметр	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, L, D, T, C	Сброшенный бит

Описание

---(R) :(Катушка сброса) Команда сброса выполняется только тогда, когда RLO предыдущей инструкции = 1 (ток поступает на катушку). При протекании тока (RLO = "1"), указанный над катушкой **<адрес>** сбрасывается в "0". При RLO = "0" (катушка не запитана) инструкция не изменяет статуса указанного операнда. В качестве **<адреса>** может также использоваться таймер (T) для сброса его значения в "0" или счетчик (C) для сброса его в "0".

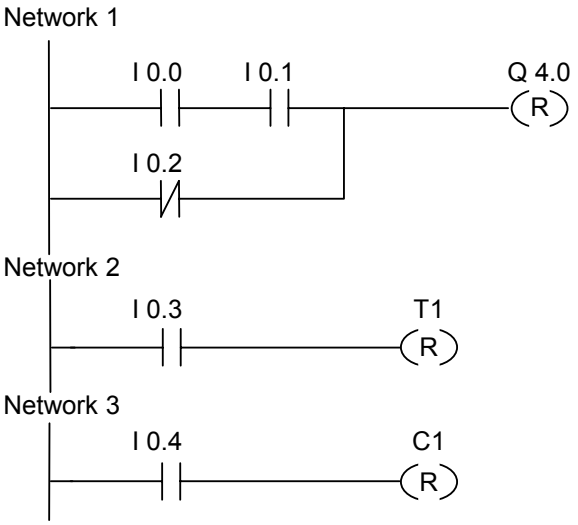
Зависимость от главного управляющего реле (Master Control Relay)

Зависимость от главного управляющего реле (MCR)проявляется только при нахождении катушки сброса внутри активированной MCR зоны. При этом если главное управляющее реле MCR включено и ток протекает через катушку сброса, адресуемый бит сбрасывается в "0". Если главное управляющее реле MCR выключено, инструкция не изменяет статуса указанного операнда, независимо от протекания тока через катушку.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	0	X	-	0

Пример



Состояние сигнала на выходе Q 4.0 сбрасывается в 0, если выполняется одно из следующих условий:

Состояние сигнала на входах I 0.0 **И** I 0.1 равно 1

ИЛИ состояние сигнала на входе I 0.2 равно 0 .

При RLO = "0", статус выхода Q4.0 остается неизменным.

Таймер T1 сбрасывается при:

состоянии сигнала "1" на входе I0.3.

Счетчик C1 сбрасывается при:

состоянии сигнала "1" на входе I0.4.

При нахождении этих цепочек внутри MCR зоны:

Если MCR включена, Q4.0, T1, и C1 сбрасываются, как в примерах выше.

Если MCR выключена, Q4.0, T1, и C1 остаются неизменными, независимо от статуса RLO (протекания тока через катушку).

1.9 ---(S) : Установка бита

Обозначение

<адрес>

---(S)

Параметр	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, L, D	Устанавливаемый бит

Описание

---(S): (Катушка установки). Инструкция **Установить бит** исполняется только тогда, когда RLO предыдущей инструкции равен 1. Если RLO равен 1, эта инструкция устанавливает указанный адрес в 1. Если RLO равен 0, то инструкция не влияет на указанный адрес, который остается неизменным.

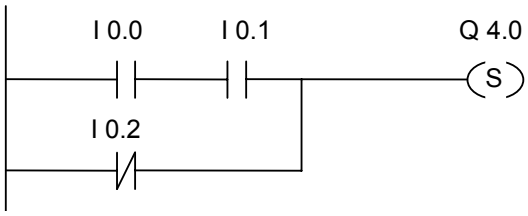
Зависимость от главного управляющего реле (Master Control Relay)

Зависимость от главного управляющего реле (MCR)проявляется только при нахождении катушки установки внутри активированной MCR зоны. При этом, если главное управляющее реле MCR включено и ток протекает через катушку установки, адресуемый бит устанавливается в "1". Если главное управляющее реле MCR выключено, инструкция не изменяет статуса указанного операнда, независимо от протекания тока через катушку.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	0	X	-	0

Пример



Состояние сигнала на выходе Q4.0 устанавливается в 1 только тогда, когда:

- равны 1 состояния сигналов на входах I0.0 И I0.1
- ИЛИ равно 0 состояние сигнала на входе I0.2

Если RLO этих цепей равен 0, то состояние сигнала на выходе Q4.0 не меняется.

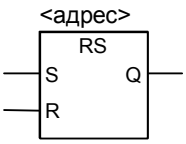
При нахождении этих цепочек внутри MCR зоны:

Если MCR включена, выход Q4.0 устанавливается как в примерах выше.

Если MCR выключена, Q4.0 остается неизменным независимо от статуса RLO (протекания тока через катушку).

1.10 RS: RS- Триггер

Обозначение



Параметр	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, L, D	Бит установки или сброса
S	BOOL	I, Q, M, L, D	Вход установки
R	BOOL	I, Q, M, L, D	Вход сброса
Q	BOOL	I, Q, M, L, D	Состояние сигнала<адрес>

Описание

Инструкция **RS- триггер** сбрасывает указанный адрес операнда, когда состояние сигнала на входе R равно 1, а состояние сигнала на входе S равно 0. Если вход R равен 0, а вход S равен 1, то триггер установлен. Если RLO на обоих входах равен 1, то триггер выполняет установку (S) или сброс (R) в соответствии с их приоритетом. RS- триггер сначала производит сброс операнда, а затем его установку, таким образом в дальнейшей программе указанный адрес будет оставаться установленным. Эти инструкции работают только тогда, когда RLO = 1. RLO, равный 0, не оказывает влияния на эти инструкции, адрес, указанный в команде остается неизменным.

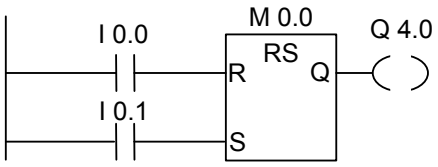
Зависимость от главного управляющего реле (Master Control Relay)

Зависимость от главного управляющего реле (MCR)проявляется только при нахождении RS-триггера внутри активированной MCR зоны. При этом, если главное управляющее реле MCR включено, то адресуемый бит устанавливается в "1" или сбрасывается в “0” как описано выше. Если главное управляющее реле MCR выключено, инструкция не изменяет статуса указанного операнда, независимо от состояния входных адресов триггера.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	x	x	x	1

Пример



Если вход I0.0= 1, а I0.1 = 0, то меркер M0.0 сброшен и выход Q4.0 равен 0. Если вход I0.0 = 0, а I0.1= 1, то меркер M0.0 установлен и выход Q4.0 = 1. Если оба сигнала равны 0, то изменения отсутствуют. Если оба сигнала равны 1, то благодаря порядку следования команд приоритетом обладает инструкция установки.

Если M 0.0 установлен ,то и Q4.0 равен 1.

При нахождении этих цепочек внутри MCR зоны:

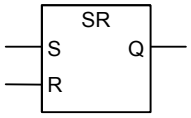
Если MCR включена, выход Q4.0 устанавливается или сбрасывается как в примерах выше.

Если MCR выключена, Q4.0 остается неизменным независимо от RLO.

1.11 SR: SR- Триггер

Обозначение

<адрес>



Параметр	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, L, D	Бит установки или сброса
S	BOOL	I, Q, M, L, D	Вход установки
R	BOOL	I, Q, M, L, D	Вход сброса
Q	BOOL	I, Q, M, L, D	Состояние сигнала<адрес>

Описание

Инструкция **SR- триггер** устанавливает указанный адрес операнда, когда состояние сигнала на входе S равно 1, а состояние сигнала на входе R равно 0. Если вход S равен 0, а вход R равен 1, то триггер сбрасывается. Если RLO на обоих входах равен 1, то триггер выполняет установку (S) или сброс (R) в соответствии с их приоритетом. SR- триггер сначала производит установку операнда, а затем его сброс, таким образом в дальнейшей программе указанный адрес будет оставаться сброшенным. Эти инструкции работают только тогда, когда RLO = 1. RLO, равный 0, не оказывает влияния на эти инструкции, адрес, указанный в команде остается неизменным.

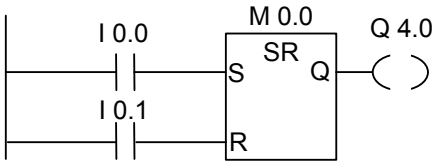
Зависимость от главного управляющего реле (Master Control Relay)

Зависимость от главного управляющего реле (MCR)проявляется только при нахождении SR-триггера внутри активированной MCR зоны. При этом, если главное управляющее реле MCR включено, то адресуемый бит устанавливается в "1" или сбрасывается в “0” как описано выше. Если главное управляющее реле MCR выключено, инструкция не изменяет статуса указанного операнда, независимо от состояния входных адресов триггера.

Биты слова состояния

	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	x	x	x	1

Пример



Если вход I0.0 = 1, а I0.1 = 0, то меркер M0.0 установлен и Q4.0 = 1.
Если I0.0= 0 а I0.1 = 1, то меркер M0.0 сброшен и Q4.0 = 0.

Если состояние обоих S/R входов равны 0, то изменения отсутствуют. Если состояния обоих входов равны 1, то благодаря порядку следования команд приоритет имеет сброс. Меркер M0.0 будет сброшен и выход Q 4.0 равен 0.

При нахождении этих цепочек внутри MCR зоны:

Если MCR включена, выход Q4.0 устанавливается или сбрасывается как в примерах выше.

Если MCR выключена, Q4.0 остается неизменным независимо от RLO.

1.12 ---(N)--- : Выделение отрицательного фронта RLO

Обозначение

<адрес>

---(N)

Параметр	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, L, D	Адрес указывает, какой бит памяти будет хранить RLO предыдущего цикла

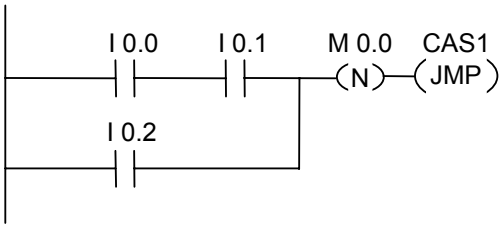
Описание

---(N)--- Инструкция **Выделение отрицательного фронта RLO** обнаруживает изменение с 1 на 0 (падающий фронт) по указанному адресу и отображает это установкой RLO в 1 после выполнения инструкции. Текущее состояние RLO сравнивается с состоянием сигнала операнда (бит памяти фронта). Если состояние сигнала операнда равно 1, а RLO перед выполнением инструкции равен 0, то RLO после выполнения инструкции будет равен 1 (импульс). Во всех остальных случаях RLO равен 0. Входной RLO затем сохраняется в указанном бите памяти.

Биты слова состояния:

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

Пример



Бит памяти фронта M0.0 сохраняет старое значение RLO . При смене значения RLO с "1" на "0", программа переходит на метку CAS1.

1.13 ---(P)---: Выделение положительного фронта RLO

<адрес>

---(P)---

Параметр	Тип данных	Область памяти	Описание
<address>	BOOL	I, Q, M, L, D	Адрес указывает, какой бит памяти будет хранить RLO предыдущего цикла

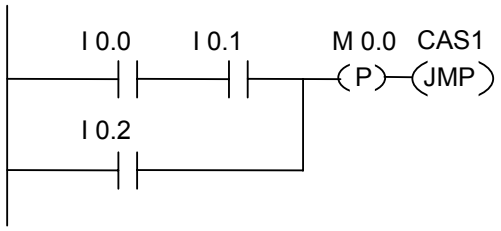
Описание

---(P)--- Инструкция **Выделение положительного фронта RLO** обнаруживает изменение с 0 на 1 (нарастающий фронт) по указанному адресу и отображает это с помощью значения RLO, равного 1, после выполнения инструкции. Текущее состояние RLO сравнивается с состоянием сигнала операнда (бит памяти фронта). Если состояние сигнала операнда равно 0, а RLO перед выполнением инструкции равен 1, то RLO будет равен 1 (импульс) после выполнения инструкции. Во всех остальных случаях RLO равен 0. Входной RLO затем сохраняется в указанном бите памяти.

Биты слова состояния:

	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	0	X	X	1

Пример



Бит памяти фронта M0.0 сохраняет старое значение RLO . При смене значения RLO с "1" на "0", программа переходит на метку CAS1.

.

1.14 ---(SAVE) Сохранить RLO в бите BR

Обозначение

---(SAVE)

Описание

---(SAVE) Команда **Сохранить RLO в бите BR** сохраняет RLO в бите BR слова состояния. Бит первичного опроса FC не сбрасывается.

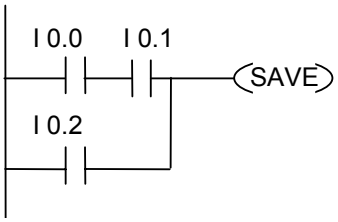
В связи с этим, если в следующем сегменте имеется логическая операция И, состояние бита BR включается в эту логическую операцию.

Не рекомендуется использовать команду “Сохранить RLO в бите BR” (LAD, FBD, STL), в следующих случаях , более подробно описанных в руководствах и встроенной справке: совместно с опросом бита BR в том же блоке или в вызываемом блоке, так как этот бит может быть изменен многими промежуточными инструкциями. Рекомендуется использовать инструкцию SAVE перед выходом из блока, так как выход ENO (= BR бит) будет установлен в определенное значение по которому Вы можете определить ошибки выполнения блока .

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	X	-	-	-	-	-	-	-	-

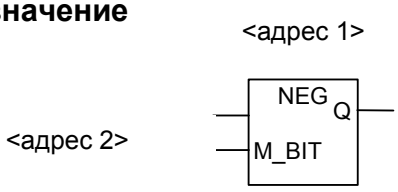
Пример



Результат логической операции (RLO) записывается в бит BR .

1.15 NEG: Выделение отрицательного фронта сигнала

Обозначение



Параметр	Тип данных	Область памяти	Описание
<адрес1>	BOOL	I, Q, M, L, D	Сигнал, контролируемый на отрицательный (падающий) фронт
<адрес2>	BOOL	I, Q, M, L, D	Адрес M_BIT указывает адрес памяти, в котором хранится предыдущее состояние адреса 1.
Q	BOOL	I, Q, M, L, D	Импульсный выход

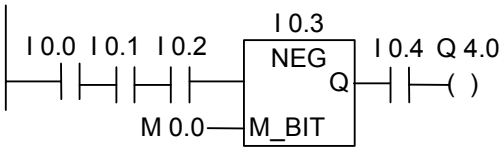
Описание

NEG Инструкция **Выделение отрицательного фронта сигнала** сравнивает состояние сигнала, который хранится в <адресе 1> с состоянием сигнала в предыдущем цикле, сохраненном в <адресе 2>. Если происходит изменение статуса сигнала с 1 на 0, то выход Q имеет значение 1, во всех остальных случаях он равен 0.

Биты слова состояния

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

Пример

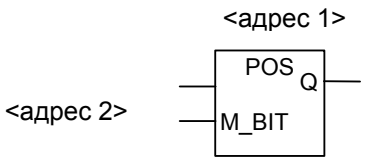


Выход Q4.0 = 1 при следующих условиях:

- состояние сигнала на входах I0.0, I0.1 и I0.2 имеет статус 1,
- И появляется падающий фронт на входе I 0.3
- И состояние сигнала на входе I0.4 равно 1.

1.16 POS: Выделение положительного фронта сигнала

Обозначение



Параметр	Тип данных	Область памяти	Описание
<адрес1>	BOOL	I, Q, M, L, D	Сигнал контролируемый на положительный (нарастающий) фронт
<адрес2>	BOOL	I, Q, M, L, D	Адрес M_указывает адрес памяти в котором хранится предыдущее состояние адреса 1.
Q	BOOL	I, Q, M, L, D	Импульсный выход

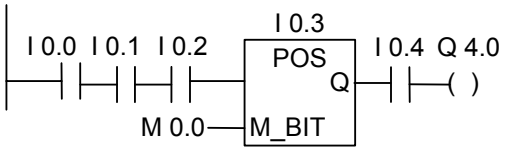
Описание

POS Инструкция **Выделение положительного фронта сигнала** сравнивает состояние сигнала в <адресе1> с предыдущим состоянием сигнала, который хранится в <адресе 1>. Если происходит изменение с 0 на 1, то выход Q имеет значение 1, во всех остальных случаях он равен 0.

Биты слова состояния

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

Пример



Выход Q4.0 = 1 при следующих условиях:

- состояние сигнала на входах I0.0, I0.1 и I0.2 имеет статус 1,
- И появляется нарастающий фронт на входе I 0.3
- И состояние сигнала на входе I0.4 равно 1.

1.17 Прямой доступ на чтение

Описание

Для прямого доступа на чтение должен быть создан сегмент, как показано ниже в примере.

Для критичных по времени приложений может возникать необходимость более частого считывания цифровой входной периферии, чем один раз за цикл выполнения OB1 . Прямой доступ на чтение предоставляет состояние цифровых входов вводного модуля на момент выполнения инструкции прямого доступа. В противном случае, Вы должны ожидать окончания выполнения следующего цикла OB1, когда будет обновлена область отображения входов новыми данными о состоянии периферии.

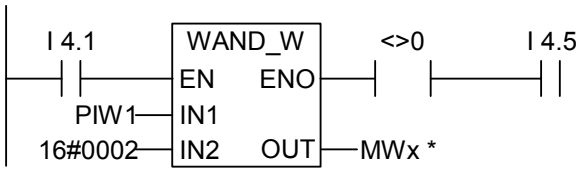
Для выполнения непосредственного чтения входной периферии используется идентификатор “PI” вместо идентификатора “I” области отображения входов. Область периферийных входов можно считывать с шириной доступа байт слово или двойное слово. Тогда как чтение отдельного бита периферии не допускается.

Условие прохождение сигнала в зависимости от статуса выполнения операции прямого доступа:

1. Необходимое слово периферийных входов считывается CPU.
2. Значение периферийных входов затем сопрягается по цифровому И с константой на втором входе и дает ненулевой результат, если входные биты установлены ("1").
3. Аккумулятор проверяется на неравенство нулю.

Пример

Контактный план для прямого доступа на чтение периферийного входа I1.1



***** MWx указывает область для сохранения полученного результата.

Описание выполнения инструкции WAND_W:

PIW1	0000000000101010
W#16#0002	0000000000000010
Result	0000000000000010

В этом примере производится опрос прямым доступом входа I1.1 последовательно с опросом входа I4.1 и I4.5.

Периферийное входное слово PIW1 содержит информацию о статусе входного бита I1.1. PIW1 сопрягается поразрядно по цифровому И с константой W#16#0002. Результат не равен 0, если I1.1 (второй бит) в периферийном байте PB2 установлен ("1"). Контакт A<>0 пропускает ток если результат выполнения инструкции WAND_W не равен 0.

1.18 Прямой доступ на запись

Описание

Для прямого доступа на запись должен быть создан сегмент, как показано ниже в примере.

Для критичных по времени приложений может возникать необходимость более частой записи цифровой выходной периферии, чем один раз за цикл выполнения OB1 . Прямой доступ на запись предоставляет возможность изменения состояния цифровых выходов выводного модуля при выполнении инструкции прямого доступа. В противном случае, Вы должны ожидать окончания выполнения следующего цикла OB1, когда будет обновлена область отображения входов новыми данными о состоянии периферии.

Для выполнения непосредственной записи выходной периферии используется идентификатор “PQ” вместо идентификатора “Q” области отображения выходов. Область периферийных выходов можно записывать с шириной доступа байт слово или двойное слово. Тогда как запись отдельного бита периферии не допускается.

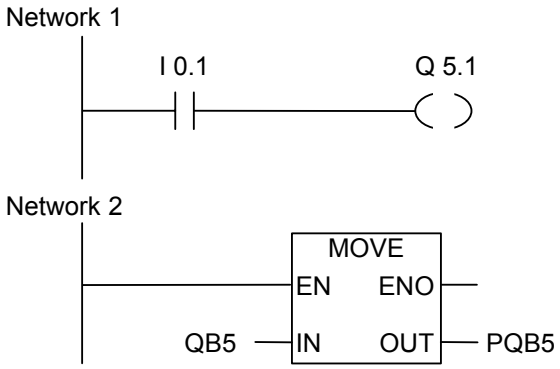
Внимание

- При записи байта выходной периферии все биты обновляются при выполнении инструкции прямого доступа к периферии .
- Если выходной бит имеет промежуточное состояние (1/0) полученное при выполнении программы, которое не должно быть передано на выходной модуль инструкция прямого доступа может привести к опасному состоянию установки (кратковременный импульс на выходе).
- Основное правило разработки: внешний выходной модуль должен получать сигнал только из одного места программы в качестве выходной катушки. Если Вы будете следовать этому правилу, большинство возможных проблем с прямым доступом будет предотвращено.

Пример

Контактный план для прямого доступа на запись периферийного выхода Q5.1

Состояние выходного бита Q байта(QB5) будет изменено или останется неизменным. Состояние сигнала на выходе Q5.1 назначается в соответствии состоянием сигнала I0.1 в сегменте 1. Содержимое QB5 копируется в соответствующую периферийную область (PQB5).



В этом примере выход Q5.1 управляет прямым доступом к периферии.

Байт PQB5 содержит значение выходного бита Q5.1.

Остальные 7 бит PQB5 будут также обновлены с помощью инструкции MOVE.

2 Инструкции сравнения

2.1 Обзор инструкций сравнения

Описание

Входы IN1 и IN2 сравниваются в соответствии с выбранным Вами типом :

== IN1 равно IN2
<> IN1 не равно IN2
> IN1 больше IN2
< IN1 меньше IN2
>= IN1 больше или равно IN2
<= IN1 меньше или равно IN2

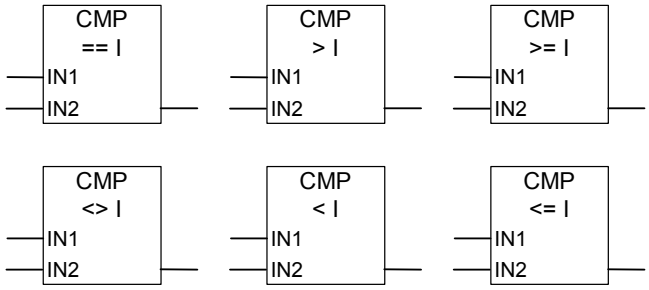
Если условие сравнения выполняется, то RLO получает значение "1". Он сопрягается с результатами опроса последующих логических операций по схеме И, если они находятся в последовательной цепи и по схеме ИЛИ в случае параллельной цепи.

Вы можете использовать следующие инструкции сравнения:

- CMP ? I : Сравнение чисел типа Integer
- CMP ? D : Сравнение чисел типа Double Integer
- CMP ? R : Сравнение чисел типа Real

2.2 CMP ? I : Сравнение чисел типа Integer

Обозначение



Параметр	Тип данных	Область памяти	Описание
Вход функции	BOOL	I, Q, M, L, D	Результат логической операции предыдущей инструкции
Выход функции	BOOL	I, Q, M, L, D	RLO выхода функции равно 1 только если вход =1 и операция сравнения выполняется и без ошибок
IN1	INT	I, Q, M, L, D или константа	Первое значение для сравнения
IN2	INT	I, Q, M, L, D или константа	Второе значение для сравнения

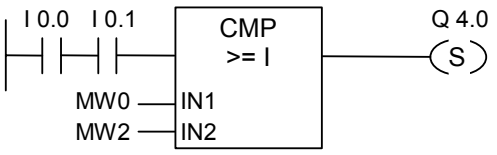
Описание

Инструкция **CMP ? I :Сравнить целые числа** может использоваться как обыкновенный контакт в любом удобном месте контактного плана. Эта инструкция сравнивает входы IN1 и IN2 в соответствии с типом сравнения, выбираемым из окна списка. Если условие сравнения выполняется, то RLO получает значение "1". Он сопрягается с результатами опроса последующих логических операций по схеме И, если они находятся в последовательной цепи и по схеме ИЛИ в случае параллельной цепи.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	x	x	x	0	-	0	x	x	1

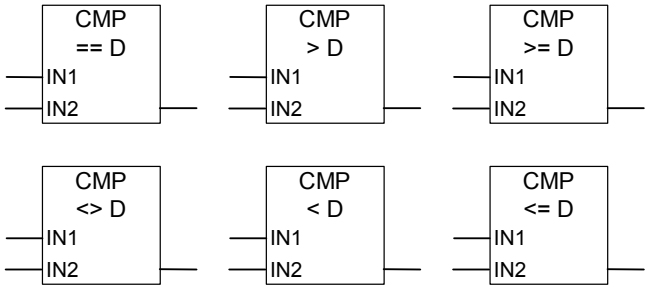
Пример



Q 4.0 устанавливается если:MW0 больше или равно MW2 и состояние сигнала на входах I0.0 и I0.1 равно 1

2.3 CMP ? D: Сравнение чисел типа Double Integer

Обозначение



Параметр	Тип данных	Область памяти	Описание
Вход функции	BOOL	I, Q, M, L, D	Результат логической операции предыдущей инструкции
Выход функции	BOOL	I, Q, M, L, D	RLO выхода функции равно 1 только если вход =1 и операция сравнения выполняется и без ошибок
IN1	DINT	I, Q, M, L, D или константа	Первое значение для сравнения
IN2	DINT	I, Q, M, L, D или константа	Второе значение для сравнения

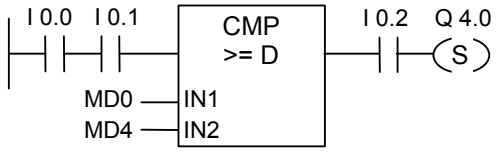
Описание

Инструкция **CMP ? D :Сравнить двойные целые числа** может использоваться как обыкновенный контакт в любом удобном месте контактного плана. Эта инструкция сравнивает входы IN1 и IN2 в соответствии с типом сравнения, выбираемым из окна списка. Если условие сравнения выполняется, то RLO получает значение "1". Он сопрягается с результатами опроса последующих логических операций по схеме И, если они находятся в последовательной цепи и по схеме ИЛИ в случае параллельной цепи.

Биты слова состояния

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

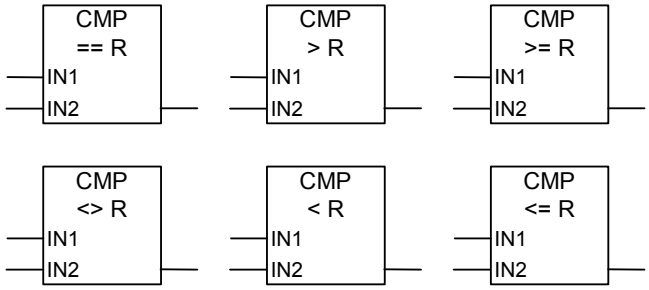
Пример



Q 4.0 устанавливается когда сигнал на входах I 0.0 и I 0.1 равен 1 и MD0 больше или равен MD4 и состояние входа I0.2 равно 1.

2.4 CMP ? R: Сравнение чисел типа Real

Обозначение



Параметр	Тип данных	Область памяти	Описание
Вход функции	BOOL	I, Q, M, L, D	Результат логической операции предыдущей инструкции
Выход функции	BOOL	I, Q, M, L, D	RLO выхода функции равно 1 только если вход =1 и операция сравнения выполняется и без ошибок
IN1	REAL	I, Q, M, L, D или константа	Первое значение для сравнения
IN2	REAL	I, Q, M, L, D или константа	Второе значение для сравнения

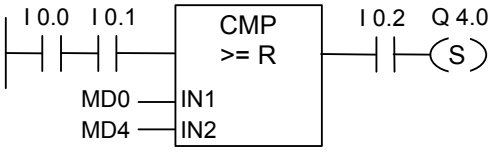
Описание

Инструкция **CMP ? R:Сравнить числа с плавающей точкой** может использоваться как обыкновенный контакт в любом удобном месте контактного плана. Эта инструкция сравнивает входы IN1 и IN2 в соответствии с типом сравнения, выбираемым из окна списка. Если условие сравнения выполняется, то RLO получает значение "1". Он сопрягается с результатами опроса последующих логических операций по схеме И, если они находятся в последовательной цепи и по схеме ИЛИ в случае параллельной цепи.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

Пример



Q 4.0 устанавливается когда сигналы на входах I 0.0 и I 0.0 равны 1 и MD0 больше или равен MD4 и сигнал на входе I 0.2 равен 1.

3 Инструкции преобразования

3.1 Обзор инструкций преобразования

Описание

Вы можете использовать следующие инструкции для преобразования двоично-десятичного кода в двоичный код и другие типы данных:

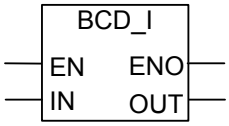
- BCD_I : Преобразование BCD- кода в Integer
- I_BCD : Преобразование Integer в BCD - код
- BCD_DI : Преобразование BCD - кода в Double Integer
- I_DI : Преобразование Integer в Double Integer
- DI_BCD : Преобразование Double Integer в BCD
- DI_R : Преобразование Double Integer в Real

- INV_I : Инверсия числа типа Integer
- INV_DI : Инверсия числа типа Double Integer
- NEG_I : Дополнительный код числа типа Integer
- NEG_DI : Дополнительный код числа типа Double Integer
- NEG_R : Инверсия знака числа типа Real

- ROUND : Округление до двойного целого
- TRUNC : Выделение целой части
- CEIL : Округление в ближайшего большего
- FLOOR: Округление до ближайшего меньшего

3.2 BCD _I : Преобразование числа в формате BCD в целое число

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	WORD	I, Q, M, L, D	Число в форматеBCD
OUT	INT	I, Q, M, L, D	Значение числа в формате Integer

Описание

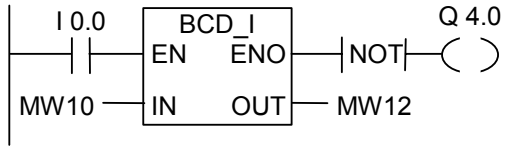
BCD_I : Инструкция преобразования **BCD в целое** считывает число в двоично-десятичном формате (BCD, ≤ 999) и преобразует это число в число с фиксированной точкой. Выходной параметр OUT содержит результат.

Выход ENO при возникновении переполнения сбрасывается в «0».

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	1	-	-	-	-	0	1	1	1

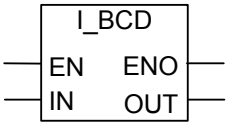
Пример



Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Содержание меркерного слова MW10 считывается как 3-разрядное число в формате BCD и преобразуется в целое число. Результат сохраняется в меркерном слове MW12. Если преобразование не выполняется, то состояние сигнала выхода Q4.0 равно 1 (ENO = EN= 0).

3.3 I_BCD : Преобразование целого числа в число в формате BCD

Обозначение



Параметры	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	INT	I, Q, M, D, L или константа	Целое число
OUT	WORD	I, Q, M, D, L	BCD значение целого числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

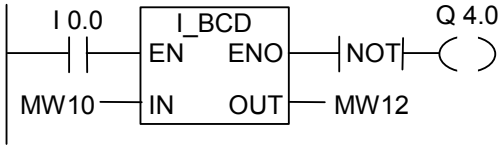
Описание

I_BCD: Инструкция преобразования **Целое в BCD** считывает содержимое входного параметра IN как целое значение и преобразует его в трехзначное число в двоично-десятичном формате (BCD, $\leq$ 999). Выходной параметр OUT содержит результат. В случае переполнения ENO сбрасывается в 0.

Биты слова состояния

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

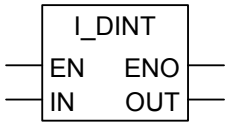
Пример



Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Содержимое меркерного слова MW10 считывается как целое число и преобразуется в формат BCD . Результат сохраняется а меркерном слове MW12. состояние сигнала на выходе Q4.0 равно 0. В случае переполнения или при состоянии сигнала на входе EN = 0 (это значит, что преобразование не выполняется), состояние сигнала на выходе Q4.0 равно 1.

3.4 I_DINT Преобразование целого числа в двойное целое

Обозначение



Параметры	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	INT	I, Q, M, D, L или константа	Целое число
OUT	DINT	I, Q, M, D, L	Двойное целое число
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

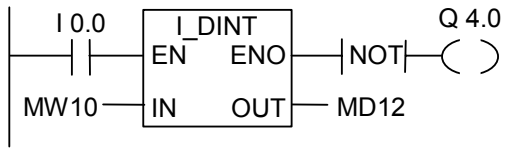
Описание

I_DINT : Инструкция преобразования **Целое в Двойное целое** считывает содержимое входного параметра IN как целое значение (16-битовое) и преобразует его в двойное целое число (32-битовое) . Выходной параметр OUT содержит результат. Выход ENO всегда имеет то же значение что и вход EN.

Биты байта состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	1	-	-	-	-	0	1	1	1

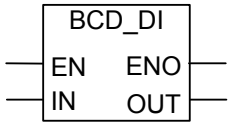
Пример



При I0.0 = "1", содержимое MW10 считывается как целое число и преобразуется в двойное целое число. Результат передается в MD12. Выход Q4.0 = "1" если преобразование не было выполнено (ENO = EN = 0).

3.5 BCD_DI : Преобразование числа в формате BCD в двойное целое число

Обозначение



Параметры	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	DWORD	I, Q, M, D, L или константа	Число в формате BCD
OUT	DINT	I, Q, M, D, L	Двойное целое значение числа BCD
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

Описание

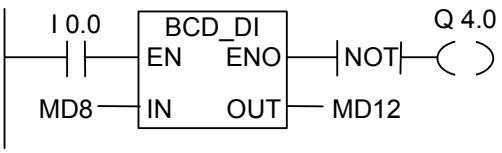
BCD_DI: Инструкция преобразования **BCD числа в двойное целое** считывает содержимое входного параметра IN как семизначное число в двоично-десятичном формате (BCD, ± 9 999 999) и преобразует это число в двойное целое число(32-битовое). Выходной параметр OUT содержит результат.

ENO всегда имеет то же состояние сигнала, что и EN.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	1	-	-	-	-	0	1	1	1

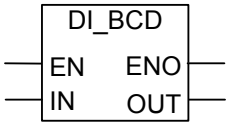
Пример



Преобразование выполняется, если состояние сигнала на входе I0.0 равно 1. Содержимое двойного меркерного слова MD8 считывается как 7-значное число в формате BCD и преобразуется в двойное целое число. Результат сохраняется в MD12. Состояние сигнала на выходе Q4.0 равно 1 если преобразование не выполняется(ENO = EN= 0).

3.6 DI_BCD : Преобразование двойного целого числа в число в формате BCD

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	DINT	I, Q, M, L, D	Двойное целое число
OUT	DWORD	I, Q, M, L, D	BCD значение двойного целого числа

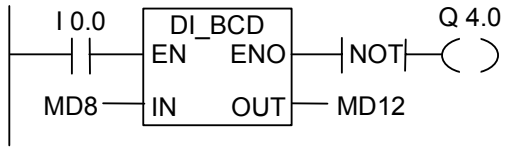
Описание

DI_BCD: Инструкция преобразования **Двойное целое в BCD** считывает содержимое входного параметра IN как двойное целое значение и преобразует его в семизначное число в формате BCD (± 9 999 999). Выходной параметр OUT содержит результат. В случае переполнения ENO сбрасывается в 0.

Биты слова состояния

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

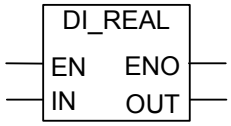
Пример



Преобразование выполняется , если состояние сигнала на I0.0 равно 1. Содержимое двойного меркерного слова MD8 считывается как двойное целое число и преобразуется в 7-значное число в формате BCD. Результат сохраняется в MD12 . Состояние сигнала на выходе Q4.0 равно 1 : если преобразование не выполняется(ENO = EN= 0) или в случае переполнения .

3.7 DI_REAL : Преобразование двойного целого числа в число с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	DINT	I, Q, M, L, D	Преобразуемая величина
OUT	REAL	I, Q, M, L, D	Результат

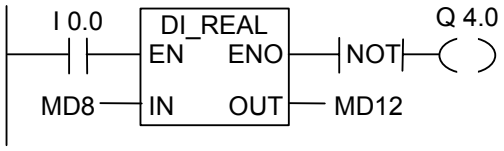
Описание

DI_REAL: Инструкция преобразования **Двойное целое в число с плавающей точкой** считывает содержимое входного параметра IN как двойное целое число и преобразует его в число с плавающей точкой. Выходной параметр OUT содержит результат. ENO всегда имеет такое же состояние сигнала, как и EN.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	1	-	-	-	-	0	1	1	1

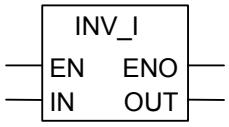
Пример



Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Содержимое двойного меркерного слова MD8 считывается как двойное целое число и преобразуется в число с плавающей точкой. Результат сохраняется в двойном меркерном слове MD12. Если преобразование не выполняется (ENO=EN=0), то состояние сигнала на выходе Q4.0 равно 1.

3.8 INV_I : Инверсия целого числа

Обозначение



Параметры	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	INT	I, Q, M, D, L	Входная величина
OUT	INT	I, Q, M, D, L	Инверсное целое число
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

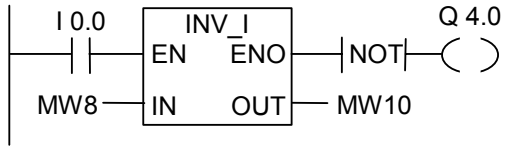
Описание

INV_I : Инструкция **Инверсия целого числа** считывает содержимое входного параметра IN и выполняет поразрядную логическую операцию Исключающее ИЛИ с маской $FFFF_H$, так что значение каждого бита инвертируется. Выходной параметр OUT содержит результат. ENO всегда имеет то же состояние сигнала, что и EN

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	1	-	-	-	-	0	1	1	1

Пример



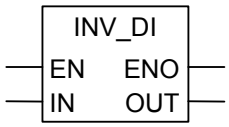
Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Значение каждого бита в MW8 инвертируется, например:

MW8 = 01000001 10000001 Результат:MW10 = 10111110 01111110

Выход Q4.0 =1 если преобразование не выполняется при ENO = EN= 0.

3.9 INV_DI : Инверсия двойного целого числа

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	DINT	I, Q, M, L, D	Входная величина
OUT	DINT	I, Q, M, L, D	Инверсия двойного целого числа

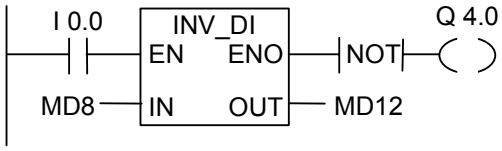
Описание

INV_DI : Инструкция **Инверсия двойного целого числа** считывает содержимое входного параметра IN и выполняет поразрядную логическую операцию Исключающее ИЛИ (см. раздел 15.6) с маской FFFF $FFFF_H$, так что значение каждого бита инвертируется. Выходной параметр OUT содержит результат. ENO всегда имеет то же состояние сигнала, что и EN.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	1	-	-	-	-	0	1	1	1

Пример



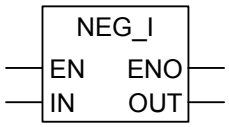
Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Значение каждого бита двойного меркерного слова MD8 инвертируется:

MD8 = F0FF FFF0 результат в: MD12 = 0F00 000F

Выход Q4.0 равен 1 , если преобразование не выполняется (ENO = EN=0).

3.10 NEG_I : Дополнительный двоичный код целого числа

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	INT	I, Q, M, L, D	Входная величина
OUT	INT	I, Q, M, L, D	Дополнительный код целого числа

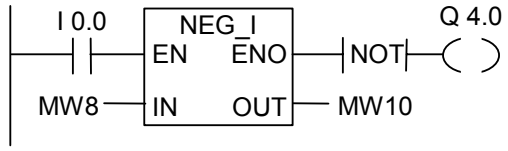
Описание

NEG_I : Инструкция **Дополнительный двоичный код целого числа** считывает содержимое входного параметра IN и изменяет знак (эквивалентно умножению числа на -1). Выходной параметр OUT содержит результат. ENO всегда имеет то же состояние сигнала, что и EN, за исключением случая, когда EN равно 1 и происходит переполнение. В этом случае состояние сигнала ENO равно 0.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

Пример



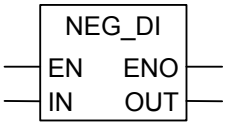
Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Значение меркерного слова MW8 выводится на OUT в меркерное словоMW10 с противоположным знаком. Пример:

MW8 = +1000 Результат: MW10 = -1000

Состояние сигнала на выходе Q4.0 равно 1 если преобразование не выполняется(ENO = EN=0). Если состояние сигнала на EN равно 1 и происходит переполнение, то ENO равно 0.

3.11 NEG_DI : Дополнительный двоичный код двойного целого числа

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	DINT	I, Q, M, L, D	Входная величина
OUT	DINT	I, Q, M, L, D	Дополнительный код двойного целого числа

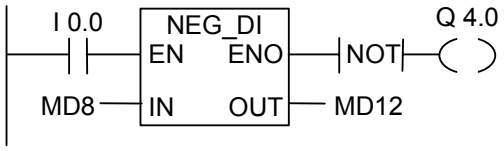
Описание

NEG_DI : Инструкция **Дополнительный двоичный код двойного целого числа** считывает содержимое входного параметра IN и изменяет знак (например, с положительного значения на отрицательное, что равноценно умножению на -1). Выходной параметр OUT содержит результат. ENO всегда имеет то же состояние сигнала, что и EN, за исключением случая, когда EN равно 1 и происходит переполнение. В этом случае состояние сигнала ENO равно 0.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

Пример



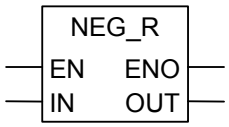
Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Значение двойного меркерного слова MD8 выводится на OUT в двойное меркерное слово MD12 с противоположным знаком

MD8 = +1000 Результат: в MD12 = -1000

Состояние сигнала на выходе Q4.0 равно 1 если преобразование не выполняется (ENO = EN=0). Если состояние сигнала на EN равно 1 и происходит переполнение, то ENO равно 0.

3.12 NEG_R : Изменение знака числа типа REAL

Обозначение



Параметры	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	REAL	I, Q, M, D, L	Входная величина
OUT	REAL	I, Q, M, D, L	Входная величина с противоположным знаком
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

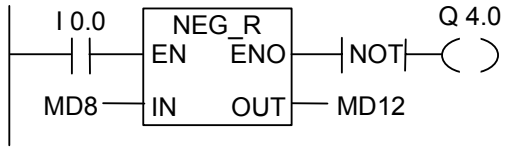
Описание

NEG_R : Инструкция **Изменение знака числа типа REAL** считывает содержимое входного параметра IN и инвертирует знаковый бит , что равносильно умножению на –1(инструкция меняет знак числа, например, с 0 для плюса на 1 для минуса). ENO всегда имеет то же состояние сигнала, что и EN.

Биты слова состояния

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

Пример

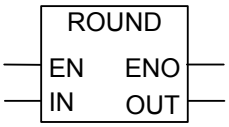


Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Значение двойного меркерного слова MD8 выводится на OUT в двойное меркерное слово MD12 с противоположным знаком, как показано в примере:

MD8 = + 6.234 Результат: MD12 = - 6.234 Если преобразование не выполняется, то состояние сигнала на выходе Q4.0 равно 1 (ENO = EN =0).

3.13 ROUND : Округление до двойного целого

Обозначение



Параметры	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	REAL	I, Q, M, D, L или константа	Округляемая величина
OUT	DINT	I, Q, M, D, L	IN округляется до ближайшего двойного целого числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

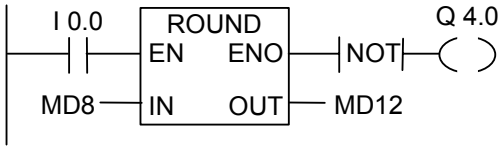
Описание

ROUND Инструкция **Округлить до двойного целого** считывает содержимое входного параметра IN как вещественное число и преобразует его в двойное целое 32-битовое число. Результат является ближайшим целым числом и содержится в выходном параметре OUT. Если дробная часть равна 0,5, то число округляется до четного числа (например, 2,5 $\rightarrow$ 2, 1,5 $\rightarrow$ 2). Если происходит переполнение, то ENO сбрасывается в 0.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	x	-	-	x	x	0	x	x	1

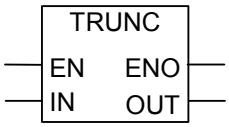
Пример



Преобразование выполняется , если I0.0 равно 1. Содержимое двойного меркерного слова MD8 считывается как вещественного число и преобразуется в двойное целое число. Результат округления до ближайшего целого сохраняется в двойном меркерном слове MD12. Состояние сигнала на выходе Q4.0 равно 1,если происходит переполнение, или если состояние сигнала на входе I0.0 равно 0 .

3.14 TRUNC : Выделение целой части из числа типа REAL

Обозначение



Параметры	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D	Округляемая величина
OUT	DINT	I, Q, M, L, D	Целая часть IN

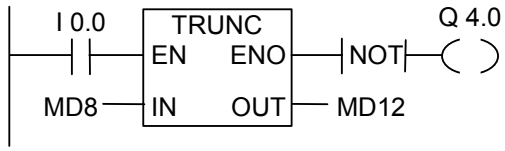
Описание

TRUNC: Инструкция **Выделение целой части из числа типа REAL** считывает содержимое входного параметра IN как вещественное число и преобразует это число в двойное целое 32-битовое число. Результат является целой частью вещественного числа. Он содержится в выходном параметре OUT. Если происходит переполнение, то ENO сбрасывается в 0.

Биты слова состояния

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

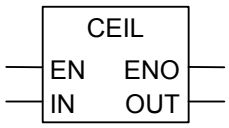
Пример



Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Содержимое двойного меркерного слова MD8 считывается как вещественное число и целая часть числа преобразованная в двойное целое сохраняется в двойном меркерном слове MD12. Состояние сигнала на выходе Q4.0 равно 1 если происходит переполнение, или если состояние сигнала на входе I0.0 равно 0 (это значит, что преобразование не выполняется).

3.15 CEIL : Округление до ближайшего большего целого числа

Обозначение



Параметры	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	REAL	I, Q, M, D, L	Преобразуемая величина
OUT	DINT	I, Q, M, D, L	Результат
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

Описание

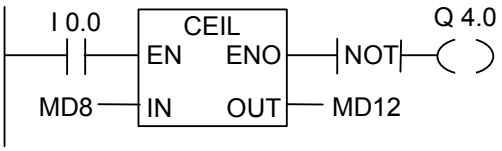
CEIL : Инструкция **Округление до ближайшего большего целого числа** считывает содержимое входного параметра IN как вещественное число и преобразует это число в двойное целое 32- битовое число . Результатом является наименьшее целое число, большее или равное заданному вещественному числу. Выходной параметр OUT содержит результат. Если происходит переполнение, то ENO равно 0.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает *:	X	-	-	X	X	0	X	X	1
Записывает **:	0	-	-	-	-	0	0	0	1

* Функция выполняется (EN = 1) ** Функция не выполняется (EN = 0)

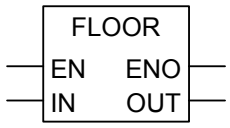
Пример



Если состояние сигнала на входе I0.0 равно 1, содержимое двойного меркерного слова MD8 считывается как вещественное число и целая часть , преобразованная в двойное целое сохраняется в двойном меркерном слове MD12. Выход Q4.0 равен 1 если происходит переполнение, или если инструкция не выполняется (вход I0.0 равен 0).

3.16 FLOOR : Округление до ближайшего меньшего целого числа

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D	Преобразуемая величина
OUT	DINT	I, Q, M, L, D	Результат

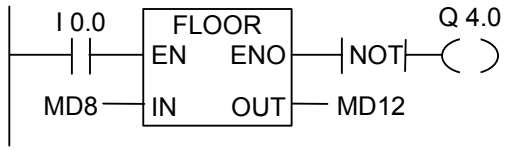
Описание

FLOOR : Инструкция **Округление до ближайшего меньшего целого числа** считывает содержимое входного параметра IN как вещественное число и преобразует это число в двойное 32-битное целое число Результатом является наибольшее целое число, меньшее или равное заданному вещественному числу. Выходной параметр OUT содержит результат. Если происходит переполнение, то ENO устанавливается в 0.

Биты слова состояния

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

Пример



Преобразование выполняется при состоянии сигнала I0.0 равном 1. Содержимое двойного меркерного слова MD8 преобразуется из числа с плавающей точкой в меньшее двойное целое число. Результат сохраняется в меркерном двойном слове MD12. При переполнении или невыполнении инструкции, выход Q4.0 устанавливается в 1.

4 Инструкции счетчиков

4.1 Обзор инструкций счетчиков

Область памяти

Счетчики имеют область, зарезервированную для них в памяти CPU. Эта область памяти резервирует по одному 16-битному слову для каждого адреса счетчика. При программировании в KOP поддерживается 256 счетчиков.

Инструкции счета являются единственными функциями, которые имеют доступ к области памяти счетчиков

Значение счетчика

Биты слова счетчика с 0 по 9 содержат значение счетчика в двоичном коде. Значение счетчика берется из аккумулятора и вводится в слово счетчика, при установке счетчика. Значение счетчика может находиться в диапазоне от 0 до 999.

Вы можете изменять значение счетчика, используя следующие инструкции:

- S_CUD : прямой/обратный счет
- S_CD : обратный счет
- S_CU : прямой счет
- ---(SC) : катушка назначения параметров
- ---(CU) : катушка прямого счета
- ---(CD) : катушка обратного счета

Структура битов в счетчике

Счетчик устанавливается на требуемое значение загрузкой числа между 0 и 999 в качестве значения счетчика, например, 127 в следующем формате: C# 127.

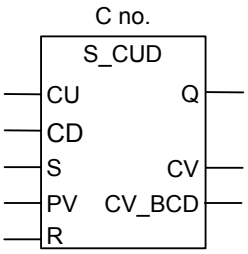
C# означает двоично-десятичный формат (BCD-формат: каждая группа из четырех битов содержит двоичный код для одного десятичного разряда).

Биты Аккумулятора с 0 по 11 содержат значение счетчика в двоично-десятичном формате. На рисунке показано содержимое аккумулятора после загрузки значения 127 и содержимое слова счетчика, после того, как он был установлен.

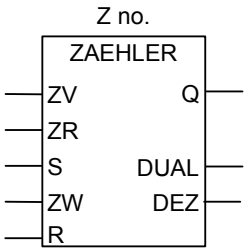
4.2 S_CUD : Назначение параметров и прямой/обратный счет

Обозначение

Английский



Немецкий



Параметры Английский	**Параметры Немецкий**	**Тип данных**	**Область памяти**	**Описание**
C no.	Z nr.	COUNTER	C	Номер счетчика .Диапазон номеров зависит от CPU.
CU	ZV	BOOL	I, Q, M, D, L	Вход прямого счета
CD	ZR	BOOL	I, Q, M, D, L	Вход обратного счета
S	S	BOOL	I, Q, M, D, L	Вход предустановки счетчика
PV	ZW	WORD	I, Q, M, D, L или константа	Задание значения счетчика от 0 до 999 как C#<значение> в формате BCD
R	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
CV	DUAL	WORD	I, Q, M, D, L	Текущее значение счетчика (целый формат)
CV_BCD	DEZ	WORD	I, Q, M, D, L	Текущее значение счетчика (формат BCD)
Q	Q	BOOL	I, Q, M, D, L	Состояние счетчика

Описание

S_CUD : (Реверсивный счетчик) устанавливается на значение, указанное на входе PV при появлении положительного фронта (изменение сигнала с 0 на 1) на входе S реверсивного счетчика. Счетчик увеличивается на 1, если состояние сигнала на входе CU изменяется с 0 на 1 (нарастающий фронт) и значение счетчика меньше 999. Счетчик уменьшается на 1, если состояние сигнала на входе CD изменяется с 0 на 1 (нарастающий фронт) и значение счетчика больше 0. При статической 1 или отрицательном фронте сигнала содержимое счетчика не изменяется. Если имеет место нарастающий фронт на обоих счетных входах, то выполняются обе инструкции и счетчик сохраняет прежнее значение. Счетчик сбрасывается, если RLO=1 появляется на входе R.

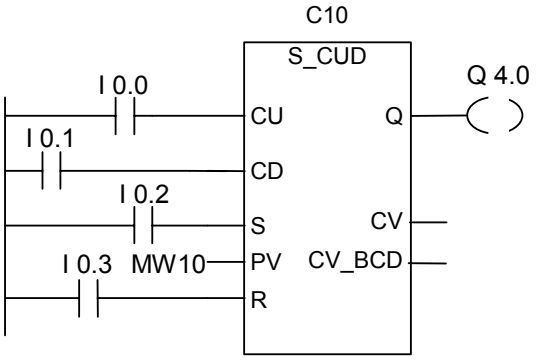
Опрос на 1 состояния сигнала на выходе Q дает 1, если значение счетчика больше 0; опрос дает результат 0, если значение счетчика равно 0.

Биты слова состояния

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

Замечание Для предотвращения ошибок счета не используйте один номер счетчика в нескольких местах программы

Пример

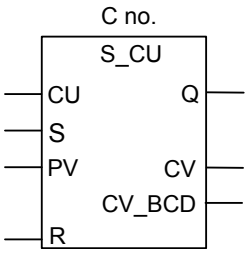


Если происходит изменение состояния сигнала с 0 на 1 на входе I0.2 счетчик C10 переустанавливается на значение из MW10. Если состояние сигнала на входе I0.0 меняется с 0 на 1, то значение счетчика C10 увеличивается на 1, кроме случая, когда значение счетчика C10 уже равно 999. Если вход I0.1 меняется с 0 на 1,то счетчик C10 уменьшается на 1 кроме случая, когда значение счетчика C10 уже равно 0. Если I0.3 меняется с 0 на 1, то значение счетчика C10 устанавливается в 0. Выход Q4.0 равен 1, когда содержимое C 10 не равно 0.

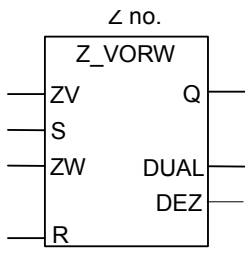
4.3 S_CU : Назначение параметров и прямой счет

Обозначение

Английский



Немецкий



Параметры Английский	Параметры Немецкий	Тип данных	Область памяти	Описание
C no.	Z no.	COUNTER	C	Номер счетчика. Диапазон номеров зависит от CPU.
CU	ZV	BOOL	I, Q, M, L, D	Вход прямого счета
S	S	BOOL	I, Q, M, L, D	Вход для предустановки счетчика
PV	ZW	WORD	I, Q, M, L, D или константа	Ввод значения счетчика в диапазоне от 0 до 999 как число в форматеBCD C#<value>
R	R	BOOL	I, Q, M, L, D	Вход сброса
CV	DUAL	WORD	I, Q, M, L, D	Текущее значение счетчика (целый формат)
CV_BCD	DEZ	WORD	I, Q, M, L, D	Текущее значение счетчика (формат BCD)
Q	Q	BOOL	I, Q, M, L, D	Состояние счетчика

Описание

S_CU : (прямой счет) устанавливается на значение, указанное на входе PV при появлении положительного фронта (изменение сигнала с 0 на 1) на входе S счетчика. Счетчик сбрасывается, если RLO=1 появляется на входе R. Счетчик увеличивается на 1, если состояние сигнала на входе CU изменяется с 0 на 1 (нарастающий фронт RLO) и значение счетчика меньше 999. При статической 1 или отрицательном фронте сигнала содержимое счетчика не изменяется.

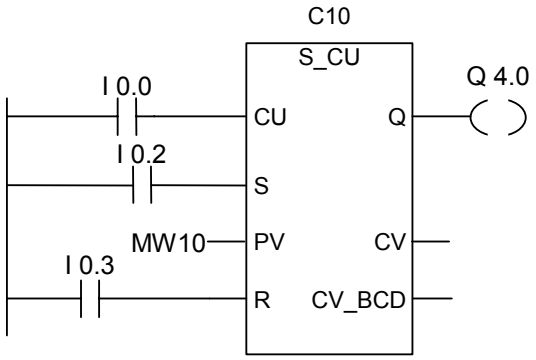
Опрос на 1 состояния сигнала на выходе Q дает 1, если значение счетчика больше 0; опрос дает результат 0, если значение счетчика равно 0.

Биты слова состояния

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

Замечание Для предотвращения ошибок не используйте один номер счетчика в нескольких местах программы

Пример

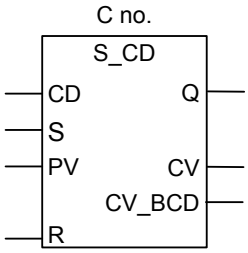


Изменение сигнала с 0 на 1 на входе I0.2 устанавливает счетчик C 10 на значение, сохраненное в MW10. Если состояние сигнала I0.0 изменяется с 0 на 1, значение счетчика C10 увеличивается на 1 до тех пор, пока не станет равным 999.Если вход I0.3 принимает значение 1, то содержимое счетчика C10 сбрасывается в 0. Состояние сигнала на выходе Q4.0 равно 1, если значение счетчика C10 отлично от 0.

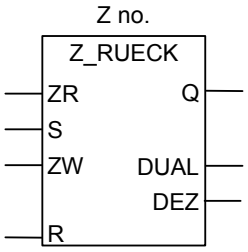
4.4 S_CD :Обратный счет

Обозначение

Английский



Немецкий



Параметры Английский	Параметры Немецкий	Тип данных	Область памяти	Описание
C no.	Z no.	COUNTER	C	Номер счетчика. Диапазон номеров зависит от CPU.
CD	ZV	BOOL	I, Q, M, L, D	Вход обратного счета
S	S	BOOL	I, Q, M, L, D	Вход для предустановки счетчика
PV	ZW	WORD	I, Q, M, L, D или константа	Ввод значения счетчика в диапазоне от 0 до 999 как число в формате BCD C#<value>
R	R	BOOL	I, Q, M, L, D	Вход сброса
CV	DUAL	WORD	I, Q, M, L, D	Текущее значение счетчика (целый формат)
CV_BCD	DEZ	WORD	I, Q, M, L, D	Текущее значение счетчика (формат BCD)
Q	Q	BOOL	I, Q, M, L, D	Состояние счетчика

Описание

S_CD : (обратный счет) устанавливается на значение, указанное на входе PV при появлении положительного фронта (изменение сигнала с 0 на 1) на входе S счетчика. Счетчик сбрасывается, если RLO=1 появляется на входе R. Счетчик уменьшается на 1, если состояние сигнала на входе CD изменяется с 0 на 1 (нарастающий фронт RLO) и значение счетчика больше 0. При статической 1 или отрицательном фронте сигнала содержимое счетчика не изменяется.

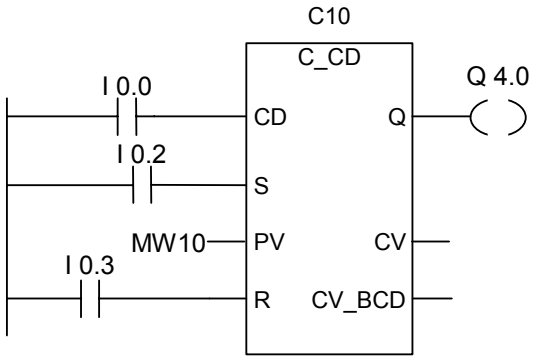
Опрос на 1 состояния сигнала на выходе Q дает 1, если значение счетчика больше 0; опрос дает результат 0, если значение счетчика равно 0.

Биты слова состояния

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

Замечание Для предотвращения ошибок не используйте один номер счетчика в нескольких местах программы

Пример



Изменение сигнала с 0 на 1 на входе I0.2 устанавливает счетчик C 10 на значение, сохраненное в MW10. Если состояние сигнала I0.0 изменяется с 0 на 1, значение счетчика C10 уменьшается на 1 до тех пор, пока не станет равным 0. Если вход I0.3 принимает значение 1, то содержимое счетчика C10 сбрасывается в 0. Состояние сигнала на выходе Q4.0 равно 1, если значение счетчика C10 отлично от 0.

4.5 ---(SC) : Установка значения счетчика

Обозначение

Английский	**Немецкий**
<C no.>	<Z no.>
---(SC)	**---(SZ)**
<Задание>	< Задание >

Параметр Английский	**Параметр Немецкий**	**Тип данных**	**Область памяти**	**Описание**
<C no.>	<Z no.>	COUNTER	C	Номер устанавливаемого счетчика
< Задание >	< Задание >	WORD	I, Q, M, L, D or constant	Значение в BCD-коде (0 - 999)

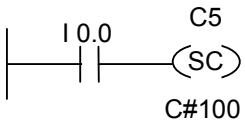
Описание

---(SC) Инструкция **Установка значения счетчика** назначает счетчику предустановленное значение при появлении положительного фронта RLO. При ее выполнении задаваемая величина передается в системную память счетчиков.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
Записывает	x	-	-	-	-	0	x	-	0

Пример



Счетчик C5 получает значение 100 если сигнал на входе I0.0 изменяется с 0 на 1 (положительный фронт RLO). Без положительного фронта RLO содержимое счетчика C5 не изменяется.

4.6 ---(CU): Счет на увеличение

Обозначение

Английский	**Немецкий**
<C no.>	<Z no.>
---(CU)	**---(ZV)**

Параметр Английский	**Параметр Немецкий**	**Тип данных**	**Область памяти**	**Описание**
<C no.>	<Z no.>	COUNTER	C	Номер счетчика; диапазон значений зависит от CPU

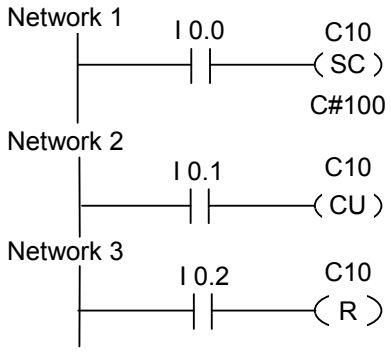
Описание

---(CU) Инструкция **Счет на увеличение** производит увеличение значения указанного счетчика на 1 при появлении нарастающего фронта RLO и значении счетчика меньше 999. Без нарастающего фронта RLO или при значении счетчика 999 ,содержимое счетчика не меняется.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
Записывает	-	-	-	-	-	0	-	-	0

Пример



При изменении сигнала на входе I0.0 с "0" на "1" (положительный фронт RLO), заданное значение 100 записывается в счетчик C10.

При изменении сигнала на входе I0.1 с "0" на "1" (положительный фронт RLO), содержимое счетчика C10 будет увеличиваться на 1 до тех пор, пока счетчик C10 не достигнет "999". Без фронта RLO, содержимое счетчика C10 не меняется.

При появлении на входе I0.2 сигнала "1", счетчик C10 сбрасывается в "0".

4.7 ---(CD): Счет на уменьшение

Обозначение

Английский	**Немецкий**
<C no.>	<Z no.>
---(CD)	**---(ZD)**

Параметр Английский	**Параметр Немецкий**	**Тип данных**	**Область памяти**	**Описание**
<C no.>	<Z no.>	COUNTER	C	Номер счетчика; диапазон значений зависит от CPU

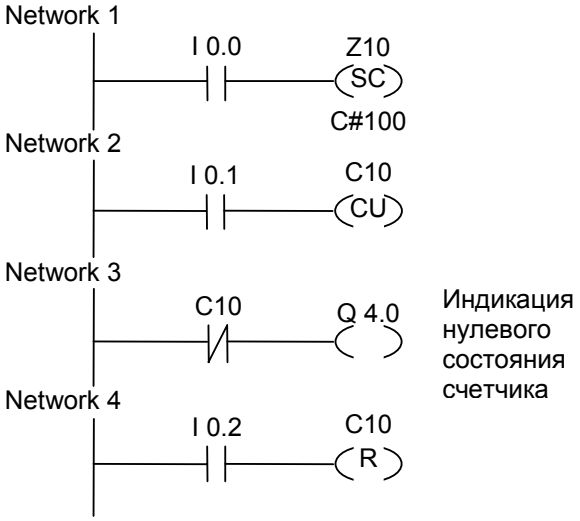
Описание

---(CD) Инструкция **Счет на уменьшение** производит увеличение значения указанного счетчика на 1 при появлении положительного фронта RLO и значении счетчика больше 0. Без нарастающего фронта RLO или при значении счетчика 0 ,содержимое счетчика не меняется.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
Записывает	-	-	-	-	-	0	-	-	0

Пример



При изменении сигнала на входе I0.0 с "0" на "1" (положительный фронт RLO), заданное значение 100 записывается в счетчик C10.

При изменении сигнала на входе I0.1 с "0" на "1" (положительный фронт RLO), содержимое счетчика C10 будет уменьшаться на 1 до тех пор, пока счетчик C10 не достигнет "0". Без фронта RLO, содержимое счетчика C10 не меняется.

При содержимом счетчике равном 0, включается выход Q4 0.

При появлении на входе I0.2 сигнала "1", счетчик C10 сбрасывается в "0".

.

5 Инструкции с блоками данных

5.1 ---(OPN) : Открыть блок данных: DB или DI

Обозначение

<DB номер.> или <DI номер>

---(OPN)

Параметр	Тип данных	Область памяти	Описание
<DB номер> <DI номер>	BLOCK_DB	DB, DI	Номер блока данных в регистре DB/DI; Допустимые значения зависят от типа CPU

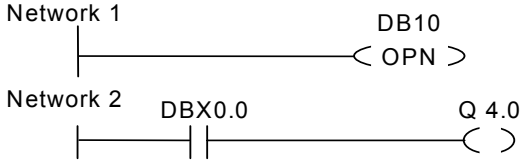
Описание

---(OPN): Вы можете использовать инструкцию **Открыть блок данных** для открытия глобального блока данных (DB) или экземплярного блока данных (DI). Инструкция **---(OPN)** является безусловным открытием блока данных. Номер блока данных заносится в регистр DB или DI. Последующие инструкции обращения к данным соответствуют обращению к номеру блока данных ранее открытого через регистры DB и DI.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	-	-	-	-

Пример



DB10 является текущим открытым блоком. Опрос бита DBX0.0 соответственно производится для бита 0 байта данных 0 блока данных DB10. Состояние сигнала этого бита присваивается выходу Q 4.0.

6 Инструкции перехода

6.1 Обзор инструкций перехода

Описание

Вы можете использовать эти инструкции во всех логических блоках , например в организационных блоках(OBs), функциональных блоках (FBs) и функциях (FCs). Возможно использование следующих инструкций перехода:

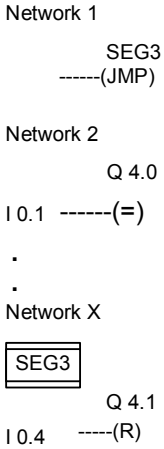
- ---(JMP)--- Безусловный переход
- ---(JMP)--- Условный переход
- ---(JMPN)--- Переход по нулю

Метка перехода как операнд

Адресом в инструкции перехода является метка . Метка перехода указывает место в программе, на которое Вам необходимо перейти. Вы вводите имя метки над блоком JMP. Имя метки состоит максимум из четырех символов. Первый символ должен быть буквой, остальные могут быть буквами или цифрами (например, SEG3).

Метка перехода как место назначения

Целевая метка вводится в начале сегмента выбором элемента LABEL из каталога элементов контактного плана. В этом блоке записывается имя метки



6.2 ---(JMP) : Безусловный переход

Обозначение

<Операнд>

---(JMP)

Описание

---(JMP) (переход на метку при RLO 1) выполняется как безусловный переход на метку если нет других элементов контактного плана между этой катушкой и питающей шиной слева (см. пример).

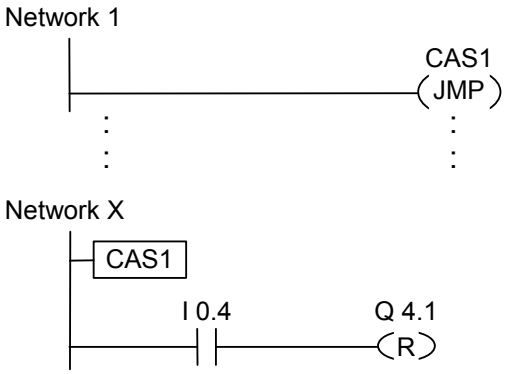
Метка перехода (LABEL) должна быть установлена для каждого перехода ---(JMP).

Инструкции между инструкцией перехода и меткой не выполняются.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	-	-	-	-

Пример



Переход всегда выполняется. Никакие инструкции между командой перехода и меткой не выполняются.

6.3 ---(JMP): Условный переход

Обозначение

<имя метки>

---(JMP)

Описание

---(JMP):Инструкция **Условный переход** (переход внутри блока при RLO 1) работает как условный переход при RLO предыдущей логической инструкции равном "1".

Целевая метка (LABEL) должна быть установлена для каждого перехода ---(JMP).

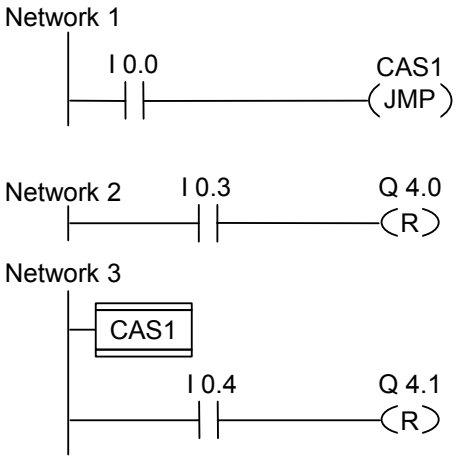
Ни одна из команд между командой перехода и меткой не выполняется.

Если условный переход не выполняется, то RLO после инструкции перехода устанавливается в "1".

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	0	1	1	0

Пример



Если состояние сигнала на входе I0.0 равно 1, переход на метку CAS1 будет выполнен. Инструкция сброса выхода Q4.0 не будет выполняться, даже если состояние сигнала I0.3 равно1.

6.4 ---(JMPN) Переход при 0

Обозначение

<имя метки>

---(JMPN)

Описание

---(JMPN): Инструкция **Переход при 0** соответствует команде “перейти на метку”, которая выполняется, если RLO равен 0.

Целевая метка (LABEL) должна устанавливаться для каждой инструкции перехода ---(JMPN).

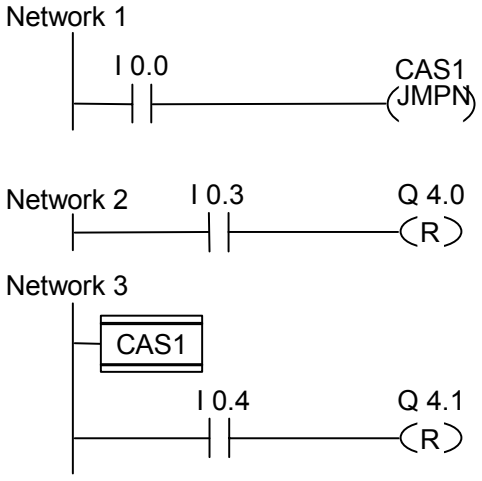
Ни одна из команд между командой перехода и меткой не выполняется.

Если условный переход не выполняется, то RLO после инструкции перехода устанавливается в "1".

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	0	1	1	0

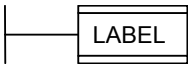
Пример



Если состояние сигнала на входе I0.0 равно 0, переход на метку CAS1 будет выполнен. Инструкция сброса выхода Q4.0 не будет выполняться, даже если состояние сигнала I0.3 равно 1.

6.5 LABEL : Метка перехода

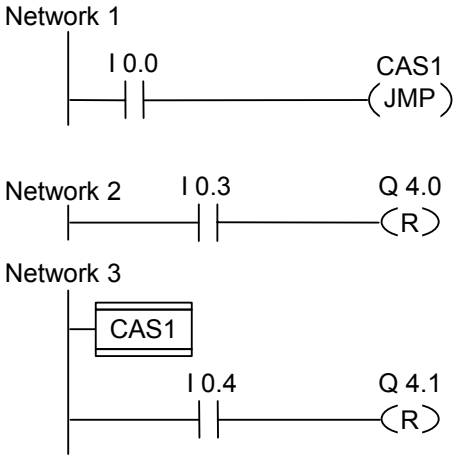
Обозначение



Описание

Метка перехода - это идентификатор места назначения инструкции перехода. Первый символ должен быть буквой алфавита, остальные могут быть цифрами. (например, CAS1) . Метка перехода должна существовать для любой инструкции перехода (**JMP** или **JMPN**).

Пример



При I0.0 = 1, переход на метку CAS1 будет выполнен.

Инструкция сброса выхода Q4.0 не будет выполняться, даже если состояние сигнала I0.3 равно1.

7 Математические инструкции с целыми числами

7.1 Обзор математических инструкций с целыми числами

Описание

Используя математические инструкции с целыми числами, Вы можете выполнять операции с двумя числами типа Integer (16 и 32 битовыми):

- ADD_I : Сложение целых чисел
- SUB_I : Вычитание целых чисел
- MUL_I : Умножение целых чисел
- DIV_I : Деление целых чисел
- ADD_DI : Сложение двойных целых чисел
- SUB_DI :Вычитание двойных целых чисел
- MUL_DI : Умножение двойных целых чисел
- DIV_DI : Деление двойных целых чисел
- MOD_DI : Получение остатка от деления двойных целых чисел

7.2 Оценка битов слова состояния при выполнении математических операций с целыми числами

Описание

Математические инструкции с целыми числами приводят к изменениям следующих бит слова состояния: CC1 и CC0, OV и OS.

Следующие таблицы показывают изменения битов слова состояния при выполнении этих инструкций с числами типа Integers (16 и 32 бит):

Допустимый диапазон значения	**CC 1**	**CC 0**	**OV**	**OS**
0 (ноль)	0	0	0	*
16 бит: -32 768 <= результат< 0 (отрицательное число) 32 бит: -2 147 483 648 <= результат< 0 (отрицательное число)	0	1	0	*
16 бит: 32 767 >= результат > 0 (положительное число) 32 бит: 2 147 483 647 >= результат > 0 (положительное число)	1	0	0	*

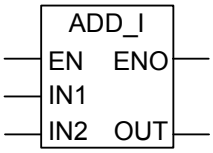
Бит OS не изменяется при выполнении этих инструкций

Недопустимый диапазон значения	**CC1**	**CC0**	**OV**	**OS**
Недопустимо малое значение (сложение) 16 бит: результат = -65536 32 бит: результат = -4 294 967 296	0	0	1	1
Недопустимо малое значение (умножение) 16 бит: результат < -32 768 (отрицательное число) 32 битs: результат < -2 147 483 648 (отрицательное число)	0	1	1	1
Переполнение (сложение , вычитание) 16 бит: результат > 32 767 (положительное число) 32 бит: результат > 2 147 483 647 (положительное число)	0	1	1	1
Переполнение (умножение, деление) 16 бит: результат > 32 767 (положительное число) 32 бит: результат > 2 147 483 647 (положительное число)	1	0	1	1
Переполнение (сложение , вычитание) 16 бит: результат < -32. 768 (отрицательное число) 32 бит: результат < -2 147 483 648 (отрицательное число)	1	0	1	1
Деление на 0	1	1	1	1

Инструкция	**CC1**	**CC0**	**OV**	**OS**
+D: результат = -4 294 967 296	0	0	1	1
/D или MOD: деление на 0	1	1	1	1

7.3 ADD_I : Сложение целых чисел

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	INT	I, Q, M, L, D или константа	Первое значение для математической инструкции
IN2	INT	I, Q, M, L, D или константа	Второе значение для математической инструкции
OUT	INT	I, Q, M, L, D	Результат операции

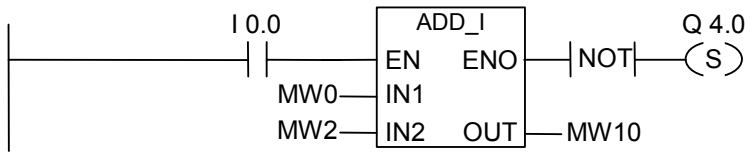
Описание

ADD_I :инструкция **Сложить целые числа** активируется при состоянии сигнала 1 на входе EN (деблокировка входа) . Эта инструкция складывает входы IN1 и IN2 и результат можно считать на выходе OUT. Если результат выходит за пределы допустимого диапазона для целых чисел, то биты OV и OS слова состояния равны 1 , а выход ENO равен 0 и таким образом, каскадное включение следующих инструкций не будет выполняться.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	x	x	x	x	x	0	x	x	1

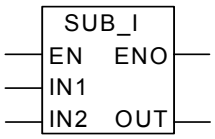
Пример



Сигнал 1 на входе I0.0 активирует блок ADD_I. Результат сложения MW0 + MW2 передается в меркерное слово MW10. Если результат выходит за пределы допустимого диапазона для целых чисел или состояние сигнала на входе I0.0 равно 0, выход Q4.0 устанавливается в 1 .

7.4 SUB_I Вычитание целых чисел

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	INT	I, Q, M, L, D или константа	Уменьшаемое
IN2	INT	I, Q, M, L, D или константа	Вычитаемое
OUT	INT	I, Q, M, L, D	Результат выполнения операции

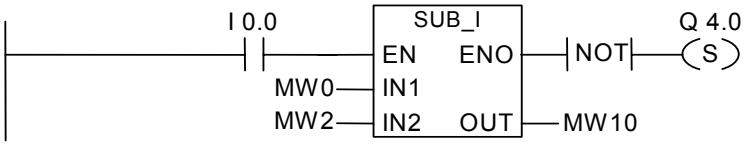
Описание

SUB_I: инструкция **Вычитание целых чисел** активируется при состоянии сигнала 1 на входе EN (деблокировка входа)**.** Эта инструкция вычитает из значения на входе IN1 значение IN2. Результат можно считать на выходе OUT. Если результат выходит за пределы допустимого диапазона для целых чисел, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

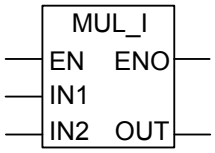
Пример



Сигнал 1 на входе I0.0 активирует блок SUB_I. Результат вычитания MW0 -MW2 передается в меркерное слово MW10. Если результат выходит за пределы допустимого диапазона для целых чисел или состояние сигнала на входе I0.0 равно 0, выход Q4.0 устанавливается в 1.

7.5 MUL_I : Умножение целых чисел

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	INT	I, Q, M, L, D или константа	Первый множитель
IN2	INT	I, Q, M, L, D или константа	Второй множитель
OUT	INT	I, Q, M, L, D	Результат выполнения операции

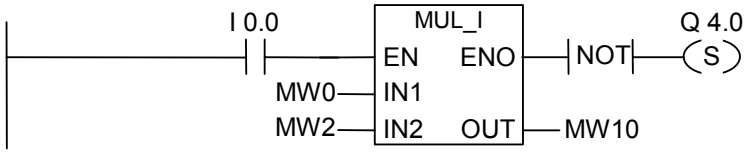
Описание

MUL_I : инструкция **Умножение целых чисел** активируется при состоянии сигнала 1 на входе EN (деблокировка входа)**.** Эта инструкция умножает значения поданные на входы IN1 и IN2, а результат можно считать на выходе OUT. Если результат выходит за пределы допустимого диапазона для целых чисел, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

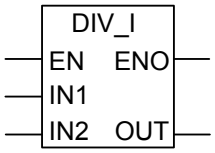
Пример



Сигнал 1 на входе I0.0 активирует блок MUL_I . Результат умножения MW0 x MW2 передается в меркерное слово MW10. Если результат выходит за пределы допустимого диапазона для целых чисел, выход Q4.0 устанавливается в 1 .

7.6 DIV_I : Деление целых чисел

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	INT	I, Q, M, L, D или константа	Делимое
IN2	INT	I, Q, M, L, D или константа	Делитель
OUT	INT	I, Q, M, L, D	Результат выполнения деления

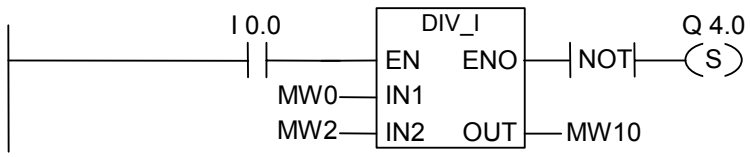
Описание

DIV_I : инструкцию **Деление целых чисел** активирует состояние сигнала 1 на входе EN (деблокировка входа)**.** Эта инструкция делит значение, поданное на вход IN1 на IN2. Результат можно считать на выходе OUT. Если результат выходит за пределы допустимого диапазона для целых чисел, то биты OV и OS слова состояния равны 1, а значение ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

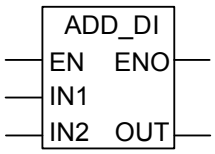
Пример



Сигнал 1 на входе I0.0 активирует блок DIV_I . Результат деления MW0 / MW2 передается в меркерное слово MW10. Если результат выходит за пределы допустимого диапазона для целых, выход Q4.0 устанавливается в 1.

7.7 ADD_DI : Сложение двойных целых чисел

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	DINT	I, Q, M, L, D или константа	Первое слагаемое
IN2	DINT	I, Q, M, L, D или константа	Второе слагаемое
OUT	DINT	I, Q, M, L, D	Результат выполнения сложения

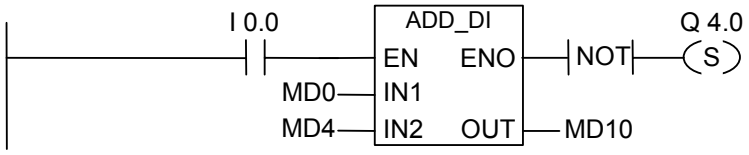
Описание

ADD_DI :Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию **Сложить двойные целые числа**. Эта инструкция складывает входы IN1 и IN2. Результат можно считать на OUT. Если результат выходит за пределы допустимого диапазона для двойных целых чисел, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	X	x	x	x	x	0	x	x	1

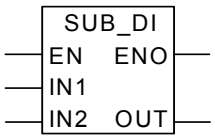
Пример



Сигнал 1 на входе I0.0 активирует блок ADD_DI . Результат сложения MD0 + MD4 передается в меркерное слово MD 10. Если результат выходит за пределы допустимого диапазона для двойных целых чисел, выход Q4.0 устанавливается в 1.

7.8 SUB_DI : Вычитание двойных целых чисел

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	DINT	I, Q, M, L, D или константа	Уменьшаемое
IN2	DINT	I, Q, M, L, D или константа	Вычитаемое
OUT	DINT	I, Q, M, L, D	Результат выполнения вычитания

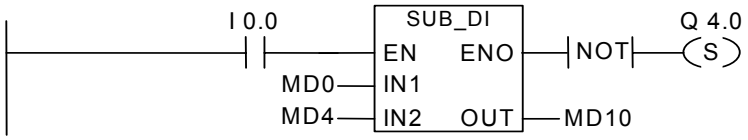
Описание

SUB_DI: Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию **Вычитание двойных целых чисел.** Эта инструкция вычитает значения поданные на входы IN1 и IN2. Результат можно считать на выходе OUT. Если результат выходит за пределы допустимого диапазона для целых чисел, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

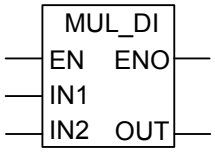
Пример



Сигнал 1 на входе I0.0 активирует блок SUB_DI. Результат вычитания MD0 –MD4 передается в меркерное двойное слово MD10. Если результат выходит за пределы допустимого диапазона для двойных целых чисел, выход Q4.0 устанавливается в 1 .

7.9 MUL_DI : Умножение двойных целых чисел

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	DINT	I, Q, M, L, D или константа	Первый множитель
IN2	DINT	I, Q, M, L, D или константа	Второй множитель
OUT	DINT	I, Q, M, L, D	Результат выполнения умножения

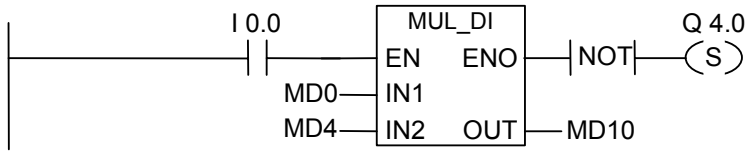
Описание

MUL_DI : инструкция **Умножение двойных целых чисел** активируется при состоянии сигнала 1 на входе EN (деблокировка входа). Эта инструкция перемножает входы IN1 и IN2. Результат в виде 32-битного целого числа можно считать на OUT. Если результат выходит за пределы допустимого диапазона для двойных целых чисел, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

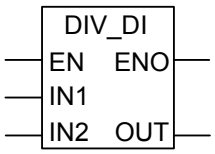
Пример



Сигнал 1 на входе I0.0 активирует блок MUL_DI . Результат умножения MD0 x MD4 передается в меркерное слово MD10. Если результат выходит за пределы допустимого диапазона для двойных целых чисел, выход Q4.0 устанавливается в 1 .

7.10 DIV_DI : Деление двойных целых чисел

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	DINT	I, Q, M, L, D или константа	Делимое
IN2	DINT	I, Q, M, L, D или константа	Делитель
OUT	DINT	I, Q, M, L, D	Целое от деления

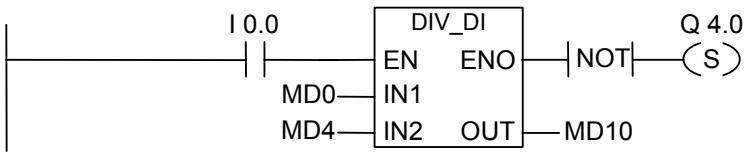
Описание

DIV_DI :Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию **Разделить двойные целые числа**. Эта инструкция делит вход IN1 на IN2. Частное от деления (округленный результат) можно считать на OUT.Инструкция Разделить двойные целые числа хранит частное от деления в виде единственного 32-битного значения в формате DINT. Эта инструкция не выдает остатка от деления. Если частное выходит за пределы допустимого диапазона для двойного целого числа, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

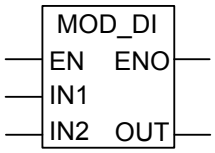
Пример



Сигнал 1 на входе I0.0 активирует блок DIV_DI . Результат деления MD0 / MD4 передается в меркерное слово MD10. Если результат выходит за пределы допустимого диапазона для двойных целых целых чисел , выход Q4.0 устанавливается в 1.

7.11 MOD_DI : Получение остатка от деления двойных целых чисел

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	DINT	I, Q, M, L, D или константа	Делимое
IN2	DINT	I, Q, M, L, D или константа	Делитель
OUT	DINT	I, Q, M, L, D	Остаток от деления

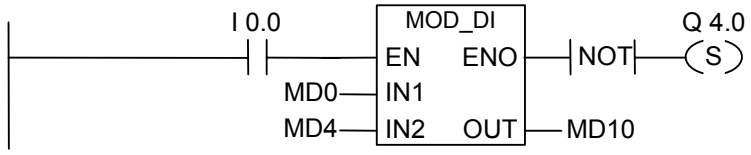
Описание

MOD_DI :Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию **получить остаток от деления двойных целых чисел**. Эта инструкция делит вход IN1 на IN2. Остаток от деления можно считать на OUT. Если результат выходит за пределы допустимого диапазона для двойного целого числа, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

Пример



Сигнал 1 на входе I0.0 активирует блок MOD_DI . Остаток от деления MD0 / MD4 передается в меркерное слово MD10. Если результат выходит за пределы допустимого диапазона для двойных целых целых чисел , выход Q4.0 устанавливается в 1 .

8 Математические инструкции с плавающей точкой

8.1 Обзор математических инструкций с плавающей точкой

Описание

Вы можете использовать математические инструкции с плавающей точкой для выполнения следующих математических операций, использующих **два 32–битных числа с плавающей точкой** (вещественный тип данных REAL) **в формате IEEE:**

- ADD_R : сложение
- SUB_R : вычитание
- MUL_R : умножение
- DIV_R : деление

Используя инструкции с плавающей точкой, Вы можете выполнять следующие операции с **одним 32-битным числом** с плавающей точкой **в формате IEEE**:

- Вычисление абсолютного значения (ABS) числа с плавающей точкой
- Вычисление квадрата (SQR) и квадратного корня (SQRT) числа с плавающей точкой
- Вычисление натурального логарифма (LN) числа с плавающей точкой
- Вычисление экспоненты числа с плавающей точкой (EXP) по основанию e (e= 2.71828...)
- Вычисление следующих тригонометрических функций угла, представленного в виде 32–битного числа с плавающей точкой:
  - синуса числа с плавающей точкой (SIN) и арксинуса числа с плавающей точкой (ASIN)
  - косинуса числа с плавающей точкой (COS) и арккосинуса числа с плавающей точкой (ACOS)
  - тангенса числа с плавающей точкой (TAN) и арктангенса числа с плавающей точкой (ATAN)

Смотрите также раздел оценки битов слова состояния.

8.2 Анализ битов слова состояния в инструкциях с плавающей точкой

Описание

Инструкции с плавающей точкой влияют на следующие биты слова состояния:

CC1 и CC0 , OV и OS .

В следующих таблицах показано влияние результатов выполнения инструкций с плавающей точкой на биты слова состояния:(32 бит):

Допустимый диапазон значений	**CC 1**	**CC 0**	**OV**	**OS**
+0, -0 (ноль)	0	0	0	*
-3.402823E+38 < результат < -1.175494E-38 (отрицательное число)	0	1	0	*
+1.175494E-38 < результат < 3.402824E+38 (положительное число)	1	0	0	*

* Бит OS не изменяется этими инструкциями.

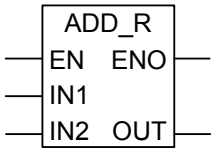
.

Недопустимый диапазон значений	**CC 1**	**CC 0**	**OV**	**OS**
Слишком малое значение -1.175494E-38 < результат < - 1.401298E-45 (отрицательное число)	0	0	1	1
Слишком малое значение +1.401298E-45 < результат < +1.175494E-38 (положительное число)	0	0	1	1
Переполнение результат < -3.402823E+38 (отрицательное число)	0	1	1	1
Переполнение результат > 3.402823E+38 (положительное число)	1	0	1	1
Недопустимое число с плавающей точкой или недопустимая инструкция (входная величина в недопустимом диапазоне)	1	1	1	1

8.3 Основные инструкции

8.3.1 ADD_R : Сложение чисел с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	REAL	I, Q, M, L, D или константа	Первое слагаемое
IN2	REAL	I, Q, M, L, D или константа	Второе слагаемое
OUT	REAL	I, Q, M, L, D	Результат сложения

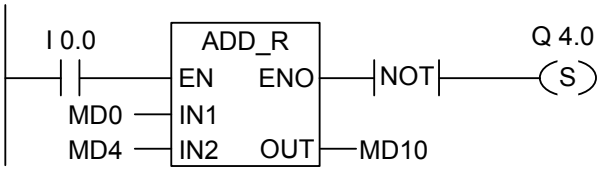
Описание

ADD_R: Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию **Сложение чисел с плавающей точкой**. Инструкция складывает входы IN1 и IN2. Результат может быть считан на выходе OUT. Если какой-либо из входов или результат не является числом с плавающей точкой, биты OV и OS устанавливаются в 1, а ENO устанавливается в 0 и, таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку битов слова состояния.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

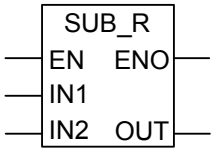
Пример



Состояние сигнала 1 на входе I0.0 активирует блок ADD_R. Результат сложения MD0 + MD4 выводится в двойное меркерное слово MD10. Если результат не является числом с плавающей точкой или если состояние сигнала на входе I0.0 равно 0, то выход Q4.0 устанавливается в 1.

8.3.2 SUB_R : Вычитание чисел с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	REAL	I, Q, M, L, D или константа	Уменьшаемое
IN2	REAL	I, Q, M, L, D или константа	Вычитаемое
OUT	REAL	I, Q, M, L, D	Результат вычитания

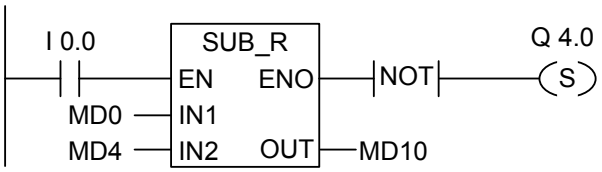
Описание

SUB_R :Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию **Вычитание чисел с плавающей точкой**. Инструкция вычитает вход IN2 из IN1. Результат может быть считан на выходе OUT. Если какой-либо из входов или результат не является числом с плавающей точкой, биты OV и OS устанавливаются в 1, а ENO устанавливается 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку битов слова состояния.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

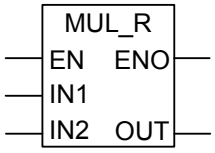
Пример



Состояние сигнала 1 на входе I0.0 активирует блок SUB_R . Результат вычитания MD0 - MD4 выводится в двойное меркерное слово MD10. Если один из входов или результат не является числом с плавающей точкой или, если состояние сигнала на входе I0.0 равно 0, то выход Q4.0 устанавливается в 1.

8.3.3 MUL_R : Умножение чисел с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	REAL	I, Q, M, L, D или константа	Первый множитель
IN2	REAL	I, Q, M, L, D или константа	Второй множитель
OUT	REAL	I, Q, M, L, D	Произведение

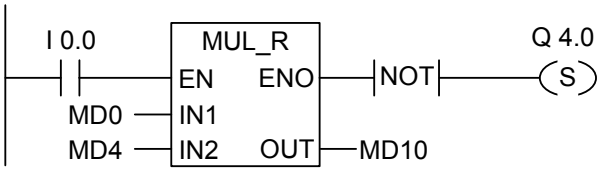
Описание

MUL_R: Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию **Умножение чисел с плавающей точкой**. Инструкция умножает вход IN1 на IN2. Результат может быть считан на выходе OUT. Если какой-либо из входов или результат не является числом с плавающей точкой, биты OV и OS устанавливаются в 1, а ENO устанавливается 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку битов слова состояния.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

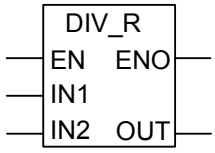
Пример



Если состояние сигнала 1 на входе I0.0 активирует блок MUL_R , результат умножения MD0 x MD4 выводится в двойное меркерное слово MD10. Если один из входов или результат не является числом с плавающей точкой или если состояние сигнала на входе I0.0 равно 0, то выход Q4.0 устанавливается в 1

8.3.4 DIV_R : Деление чисел с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN1	REAL	I, Q, M, D, L или константа	Делимое
IN2	REAL	I, Q, M, D, L или константа	Делитель
OUT	REAL	I, Q, M, D, L	Результат деления
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

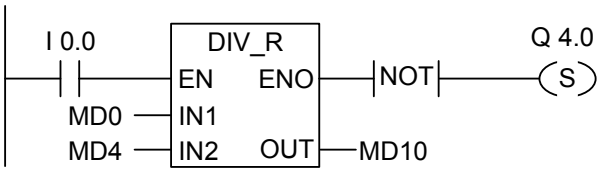
Описание

DIV_R : Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию **Деление вещественных чисел**. Инструкция делит вход IN1 на IN2. Результат может быть считан на выходе OUT. Если какой-либо из входов или результат не является числом с плавающей точкой, биты OV и OS устанавливаются в 1, а ENO устанавливается в0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку битов слова состояния.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

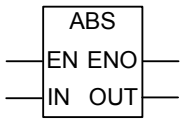
Пример



Состояние сигнала 1 на входе I0.0 активирует блок DIV_R. Результат деления MD0 на MD4 выводится в двойное меркерное слово MD10. Если один из входов или результат не является числом с плавающей точкой или если состояние сигнала на входе I0.0 равно 0, то выход Q4.0 устанавливается в 1.

8.3.5 ABS: Вычисление модуля числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Число
OUT	REAL	I, Q, M, L, D	Абсолютное значение

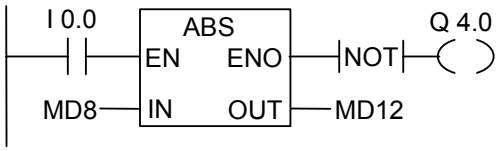
Описание

ABS: С помощью инструкции **Вычисление модуля числа с плавающей точкой,** Вы можете определить абсолютную величину числа с плавающей точкой.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	1	-	-	-	-	0	1	1	1

Пример



Если I0.0 = 1, то абсолютное значение MD8 выводится на MD12.

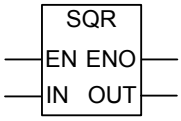
MD8 = +6.234 дает в результате MD12 = 6.234.

Выход Q4.0 равен 1 , если преобразование не выполняется (ENO = EN = 0).

8.4 Дополнительные инструкции

8.4.1 SQR: Вычисление квадрата числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Число
OUT	REAL	I, Q, M, L, D	Квадрат числа

Описание

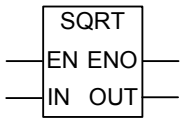
SQR :С помощью инструкции **Вычисление квадрата числа с плавающей точкой,** Вы можете возвести число с плавающей точкой в квадрат. Смотрите также оценку битов слова состояния.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	x	x	x	x	x	0	x	x	1

8.4.2 SQRT: Вычисление квадратного корня числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Число
OUT	REAL	I, Q, M, L, D	Квадратный корень из числа

Описание

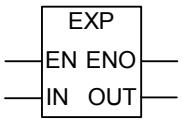
SQRT: С помощью инструкции **Вычисление квадратного корня числа плавающей точкой,** Вы можете извлечь квадратный корень из числа с плавающей точкой. Инструкция возвращает положительный результат, если значение входного значения больше 0. Единственное исключение для –0: результат также –0. Смотрите также оценку битов слова состояния.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

8.4.3 EXP : Вычисление экспоненты числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Число
OUT	REAL	I, Q, M, L, D	Экспоненциальное значение числа

Описание

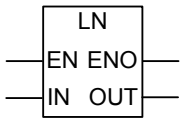
С помощью инструкции **Вычисление экспоненты числа с плавающей точкой** , Вы можете найти экспоненциальное значение числа с плавающей точкой по основанию e (=2,71828...). Смотрите также оценку битов слова состояния.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

8.4.4 LN : Вычисление натурального логарифма числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Число
OUT	REAL	I, Q, M, L, D	Натуральный логарифм числа

Описание

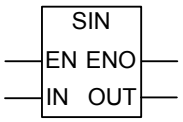
С помощью инструкции **Вычисление натурального логарифма числа с плавающей точкой** ,Вы можете определить натуральный логарифм числа с плавающей точкой. Смотрите также оценку битов слова состояния.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

8.4.5 SIN : Вычисление синуса числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Число с плавающей точкой
OUT	REAL	I, Q, M, L, D	Выходная величина: Синус числа с плавающей точкой

Описание

Инструкция SIN вычисляет синус числа с плавающей точкой. Это число должно представлять значение угла в радианах.

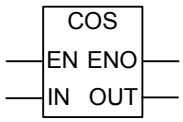
Смотрите также оценку битов слова состояния.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	x	x	x	x	x	0	x	x	1

8.4.6 COS: Вычисление косинуса числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Входная величина: число с плавающей точкой
OUT	REAL	I, Q, M, L, D	Выходная величина: Косинус числа с плавающей точкой

Описание

Инструкция COS вычисляет синус числа с плавающей точкой. Это число должно представлять значение угла в радианах.

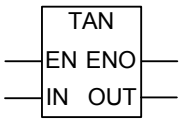
Смотрите также оценку битов слова состояния.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

8.4.7 TAN: Вычисление тангенса числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Входная величина: число с плавающей точкой
OUT	REAL	I, Q, M, L, D	Выходная величина: Тангенс числа с плавающей точкой

Описание

Инструкция TAN вычисляет тангенс числа с плавающей точкой. Это число должно представлять значение угла в радианах.

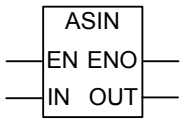
Смотрите также оценку битов слова состояния

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

8.4.8 ASIN: Вычисление арксинуса числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Входная величина: число с плавающей точкой
OUT	REAL	I, Q, M, L, D	Выходная величина: Арксинус числа с плавающей точкой

Описание

Инструкция ASIN вычисляет арксинус числа с плавающей точкой в пределах допустимых значений от –1 до 1. Результат представляется как значение угла в радианах в диапазоне: $-\pi/2 \leq$ Выходная величина $\leq \pi/2$, где π =3.1415

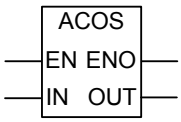
Смотрите также оценку битов слова состояния

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

8.4.9 ACOS: Вычисление арккосинуса числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Входная величина: число с плавающей точкой
OUT	REAL	I, Q, M, L, D	Выходная величина: Арккосинус числа с плавающей точкой

Описанеие

Инструкция ACOS вычисляет арккосинус числа с плавающей точкой в пределах допустимых значений от –1 до 1. Результат представляется как значение угла в радианах в диапазоне: $0 \leq$ Выходная величина $\leq +\pi$, где $\pi = 3.1415$

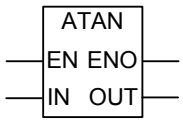
Смотрите также оценку битов слова состояния **.**

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	x	x	x	x	x	0	x	x	1

8.4.10 ATAN: Вычисление арктангенса числа с плавающей точкой

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	REAL	I, Q, M, L, D или константа	Входная величина: число с плавающей точкой
OUT	REAL	I, Q, M, L, D	Выходная величина: Арктангенс числа с плавающей точкой

Описание

Инструкция ATAN вычисляет арктангенс числа с плавающей точкой . Результат представляется как значение угла в радианах в диапазоне: $-\pi/2 \leq$ Выходная величина $\leq +\pi/2$, где $\pi = 3.1415$

Смотрите также оценку битов слова состояния .

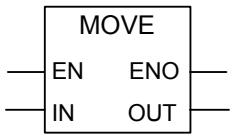
Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	x	x	x	x	x	0	x	x	1

9 Инструкции передачи

9.1 MOVE : Передача значения

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN	Все элементарные типы данных 8,16 или 32-битовые	I, Q, M, L, D или константа	Исходная область
OUT	Все элементарные типы данных 8,16 или 32-битовые	I, Q, M, L, D	Целевая область

Описание

Инструкция **MOVE** (Передать значение) активируется при разрешении на входе EN. Значение, указанное на входе IN, копируется в адрес, указанный на выходе OUT. ENO имеет то же состояние сигнала, что и EN. Инструкция MOVE позволяет передавать данные с шириной доступа BYTE, WORD или DWORD. Типы данных, определенные пользователем, такие как массивы или структуры, должны копироваться с помощью системной функции ”BLKMOV” (SFC 20).

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	1	-	-	-	-	0	1	1	1

MCR : Влияние главного управляющего реле

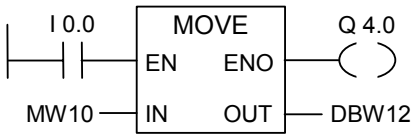
Влияние главного управляющего реле (MCR) проявляется только в случае, если инструкция Move находится внутри MCR зоны. В этом случае, внутри зоны MCR, если функция MCR включена и разрешающий сигнал подан на вход EN; данные будут скопированы , как описано ранее. Если функция MCR выключена, то при выполнении функции MOVE , значение "0" будет записано в указанный адрес на выходе OUT независимо от содержимого входа IN .

Замечание

При передаче значений между переменными различной ширины доступа, старшие байты отсекаются или заполняются нулями:

Пример: Двойное слово	**1111 1111**	**0000 1111**	**1111 0000**	**0101 0101**
Функция MOVE	**Результат**			
В двойное слово:	1111 1111	0000 1111	1111 0000	0101 0101
В байт:				0101 0101
В слово:			1111 0000	0101 0101
Пример: Байт :				**1111 0000**
Функция MOVE	**Результат**			
В байт:				1111 0000
В слово:			0000 0000	1111 0000
В двойное слово:	0000 0000	0000 0000	0000 0000	1111 0000

Пример



Инструкция выполняется, если вход I0.0 = 1. Содержимое MW10 копируется в слово данных 12 открытого блока данных DB.

Если инструкция выполняется, выход Q4.0 устанавливается в 1.

Если этот пример находится внутри активированной зоны MCR:

- При включенной функции MCR, MW10 копируется в DBW12.
- При выключенной функции MCR, значение "0" записывается в DBW12.

10 Команды управления программой

10.1 Обзор команд управления программой

Описание

Следующие команды управления программой доступны пользователю:

- ---(CALL) Вызов FC/SFC без параметров
- CALL_FB : Вызов блока FB в графическом виде
- CALL_FC : Вызов блока FC в графическом виде
- CALL_SFB : Вызов системного FB в графическом виде
- CALL_SFC : Вызов системной FC в графическом виде
- Вызов мультиэкземпляра
- Вызов блока из библиотеки
- Команды Master Control Relay(Главное реле управления)
- Правила использования функций Master Control Relay MCR
- ---(MCR<) Включить Master Control Relay
- ---(MCR>) Выключить Master Control Relay
- ---(MCRA) Активировать зону Master Control Relay
- ---(MCRD) Деактивировать зону Master Control Relay
- RET Возврат

10.2 ---(Call) : Вызов FC (SFC) без параметров

Обозначение

<FC/SFC no.>

---(CALL)

Параметр	Тип данных	Область памяти	Описание
<FC/SFC no.>	BLOCK_FC BLOCK_SFC	-	Номер FC/SFC; диапазон номеров зависит от типа CPU

Описание

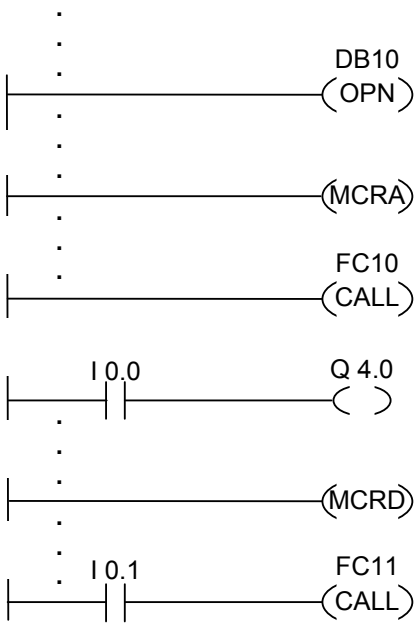
---(Call) С помощью инструкции **Вызвать FC/SFC без параметров** можно вызвать функцию (FC) или системную функцию (SFC), которые не имеют параметров. Вызов выполняется только при подаче RLO 1 на управляющую катушку CALL. Если инструкция выполняется, то выполняются следующие функции:

- Сохраняется адрес, необходимый для возврата в вызывающий блок.
- Предыдущая область локальных данных заменяется текущей областью локальных данных.
- Бит MA (бит активации MCR) записывается в стек блоков (BSTACK).
- Для вызываемой FC или SFC создается новая область локальных данных.

Биты слова состояния

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Условный вызов	Записывает	-	-	-	-	0	0	1	-	0
Безусловный вызов	Записывает	-	-	-	-	0	0	1	1	0

Пример



На рисунке выше показана часть пользовательской программы, написанной в контактном плане. В теле данного функционального блока открывается блок данных DB10 и активируется зона MCR . Если выполняется безусловный вызов FC10, то выполняются следующие функции:

Сохраняется адрес, необходимый для возврата в текущий FB, а также регистры для DB10 и для экземпляра блока данных FB.

Помещается в стек блоков (BSTACK) бит MA, установленный в 1 в команде MCRA , и этот бит сбрасывется в 0 для вызванного FC10.

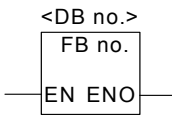
Исполнение программы продолжается в FC10. Если Вы хотите использовать функцию MCR в FC10, Вы должны ее там повторно ативировать. Когда FC10 завершается, выполнение программы продолжается в вызывающем FB. Бит MA восстанавливается, DB10 и экземпляр блока данных пользовательского FB снова являются текущими DB независимо от того, какие DB были использованы в FC10. После возврата из FC10 состояние сигнала входа. I0.0 присваивается выходу Q4.0. Вызов FC11 является условным. Он выполняется только тог да, когда состояние входа I0.1 равно 1. Если вызов выполняется, то происходит то же самое, что и при вызове FC10.

Замечание

После возврата в вызывающий блок, ранее открытый блок данных остается открытым не во всех случаях. Более подробную информацию Вы найдете в файле README .

10.3 CALL_FB : Вызов FB в графическом виде

Обозначение



Символьные имена интерфейса зависят от вызываемого блока (сколько и какие параметры в нем описаны). EN, ENO и имя или номер блока FB обязательны.

Параметры	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входов
ENO	BOOL	I, Q, M, L, D	Деблокировка выходов
FB no. DB no.	BLOCK_FB BLOCK_DB	- -	Номера блоков FB/DB, диапазон номеров зависит от CPU

Описание

CALL_FB (Вызов FB в графическом виде) производится при состоянии сигнала на входе EN = 1. При выполнении команды CALL_FB выполняются следующие процедуры:

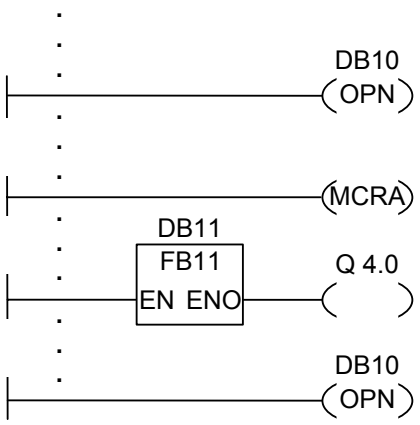
- Сохранение адреса возврата в вызывающий блок,
- Сохранение значений регистров блоков данных (DB и DI),
- Бит MA (бит активности функции MCR) сохраняется в BSTACK
- Создание новой области в локальном стеке для вызываемого функционального блока и ее активация вместо локальной области вызывающего блока.

После этого выполнение программы продолжается в вызванном функциональном блоке. Для управления выходом ENO используется оценка бита BR. Пользователь должен сохранять необходимый статус (сообщение об ошибке) в бите BR с помощью инструкции ---(SAVE).

Биты слова состояния

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Условный вызов	Записывает	X	-	-	-	0	0	X	X	X
Безусловный вызов	Записывает	-	-	-	-	0	0	X	X	X

Пример



Сегменты на рисунке являются частью пользовательского функционального блока . В этом блоке открывается блок DB10 и активируется функция MCR. Если выполняется безусловный вызов FB11, происходит следующее:

Сохраняются точка возврата в вызывающий блок и регистр блоков данных для DB10 и экземплярного блока вызывающего FB. Бит MA , установленный в 1 инструкцией MCRA сохраняется в стеке блоков BSTACK и сбрасывается в 0 для вызванного блока FB11. Выполнение программы продолжается в FB11. Если FB11 требуется активация MCR, то это должно быть повторно сделано в функциональном блоке. Значение RLO должно быть сохранено в бите BR с помощью инструкции [SAVE] для оценки ошибки в блоке FB. После выполнения FB11, программа возвращается для выполнения в вызывающий FB. Бит MA восстанавливается, как и номер экземплярного блока данных , восстанавливаемый в регистре DI. Если FB11 обработан без ошибок, состояние сигнала на выходе ENO = 1 и ,соответственно, выход Q4.0 также равен 1.

Замечание

После вызова блоков FB/SFB, номер ранее открытого блока данных будет утерян. Требуемый блок данных нужно снова открыть в вызывающем блоке.

10.4 CALL_FC Вызов функции в графическом виде

Обозначение

Символьные имена интерфейса зависят от вызываемого блока (сколько и какие параметры в нем описаны). EN, ENO и имя или номер функции обязательны.

Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
FC no.	BLOCK_FC	-	Номер функции, диапазон номеров зависит от CPU

Описание

CALL_FC (вызов функции в графическом виде) вызывает для выполнения FC при состоянии сигнала на входе EN = 1. При этом выполняются следующие процедуры:

- Сохранение адреса возврата в вызывающий блок,
- Создание новой области в локальном стеке для вызываемой функции и замена локальных данных вызывающего блока на текущие данные.
- Бит MA (бит активности функции MCR) сохраняется в BSTACK,

После этого, выполнение программы продолжается в вызванном блоке.

Бит BR позволяет организовать управление выходом ENO. Пользователь должен назначить необходимый статус этого бита (индикация ошибки) в вызываемом блоке с помощью команды **---(SAVE).**

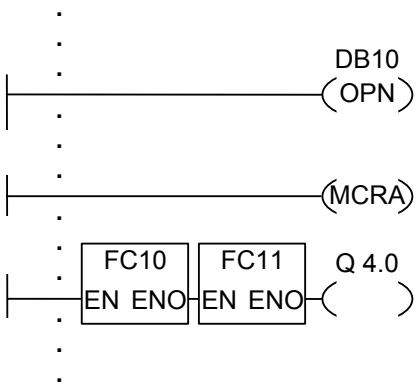
Если в вызываемой функции в таблице описаний созданы параметры типа IN, OUT и IN_OUT, эти переменные появляются в качестве формальных параметров при вызове этого блока.

При вызове функции, Вы **должны** назначать фактические параметры для всех формальных параметров вызываемой функции. Задание начальных значений для параметров функции не допускается.

Биты слова состояния

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Условный вызов	Записывает	X	-	-	-	0	0	X	X	X
Безусловный вызов	Записывает	-	-	-	-	0	0	X	X	X

Пример



Сегменты на рисунке являются частью пользовательского функционального блока, написанного на языке контактного плана . В этом блоке открывается блок DB10 и активируется функция MCR. Если выполняется безусловный вызов FC11, происходит следующее:

Сохраняются точка возврата в вызывающий блок и регистры блоков данных для DB10 и экземплярного блока вызывающего FB. Бит MA , установленный в 1 инструкцией MCRA сохраняется в стеке блоков BSTACK и сбрасывается в 0 для вызванного блока FC10. Выполнение программы продолжается в FC10. Если FC10 требуется активация MCR, то это должно быть повторно сделано в самой функции. Значение RLO должно быть сохранено в бите BR с помощью инструкции---(SAVE) для возможности оценки ошибки в блоке FB. После выполнения FC10, программа возвращается для выполнения в вызывающий FB. Бит MA восстанавливается, как и номер экземплярного блока данных , восстанавливаемый в регистре DI. Если FC10 обработан без ошибок, дальнейшая обработка продолжается в зависимости от состояния выхода ENO:

Если состояние сигнала на выходе ENO = 1, производится обработка блока FC11.

Если состояние сигнала на выходе ENO = 0, производится обработка следующего сегмента.

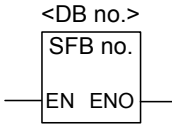
Если FC11 обработан без ошибок то выход ENO и Q4.0 также равны 1.

Замечание

После возврата программы в вызывающий блок, не всегда гарантируется , что номер ранее открытого блока данных будет восстановлен. Для справки обратитесь к файлу README.

10.5 CALL_SFB : Вызов системного FB в графическом виде

Обозначение



Символьные имена интерфейса зависят от вызываемого системного блока (сколько и какие параметры в нем описаны). EN, ENO и имя или номер блока SFB обязательны.

Параметры	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входов
ENO	BOOL	I, Q, M, L, D	Деблокировка выходов
SFB no. DB no.	BLOCK_SFB BLOCK_DB	- -	Номера блоков SFB/DB ; диапазон номеров зависит от CPU

Описание

CALL_SFB (Вызов SFB в графическом виде) выполняется при состоянии сигнала на входе EN = 1. При исполнении CALL_SFB выполняются следующие процедуры:

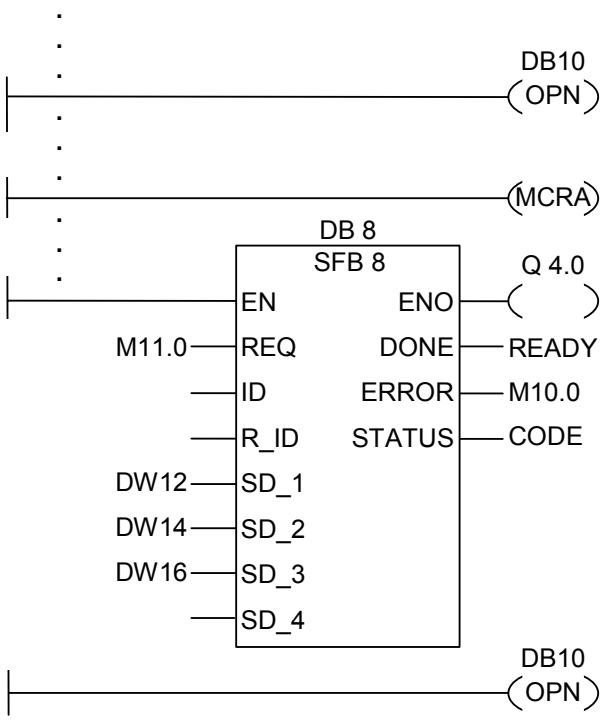
- Сохранение адреса возврата в вызывающий блок,
- Сохранение значений регистров блоков данных (DB и DI),
- Создание новой области в локальном стеке для вызываемого функционального блока и ее активация .
- Бит MA (бит активности функции MCR) сохраняется в BSTACK

После этого, выполняется программа вызываемого SFB. Параметр ENO = 1 если блок был обработан без ошибок .

Биты слова состояния

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Условный вызов	Записывает	X	-	-	-	0	0	X	X	X
Безусловный вызов	Записывает	-	-	-	-	0	0	X	X	X

Пример



Сегменты на рисунке являются частью пользовательского функционального блока . В этом блоке открывается блок DB10 и активируется функция MCR. При безусловном вызове SFB8, происходит следующее:

Сохраняются точка возврата в вызывающий блок и регистры блоков данных для DB10 и экземплярного блока вызывающего FB. Бит MA , установленный в 1 инструкцией MCRA сохраняется в стеке блоков BSTACK и сбрасывается в 0 для вызванного блока SFB8. Выполнение программы продолжается в SFB8. После завершения обработки SFB8 , программа возвращается в вызывающий блок. Восстанавливаются бит MA и номер экземплярного блока данных пользовательского FB. Если SFB8 обработан без ошибок, состояние сигнала на выходе ENO = 1 и, соответственно, выход Q4.0 также равен 1.

Замечание

После вызова блоков FB/SFB, номер ранее открытого блока данных будет утерян. Требуемый блок данных нужно снова открыть в вызывающем блоке.

10.6 CALL_SFC Вызов системной функции FC в графическом виде

Обозначение

Символьные имена интерфейса зависят от вызываемого блока (сколько и какие параметры в нем описаны). EN, ENO и имя или номер SFC обязательны .

Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
SFC no.	BLOCK_SFC	-	Номер SFC, диапазон номеров зависит от CPU

Описание

CALL_SFC (Вызов SFC в графическом виде) выполняется при состоянии сигнала на входе EN = 1. При исполнении CALL_SFC выполняются следующие процедуры:

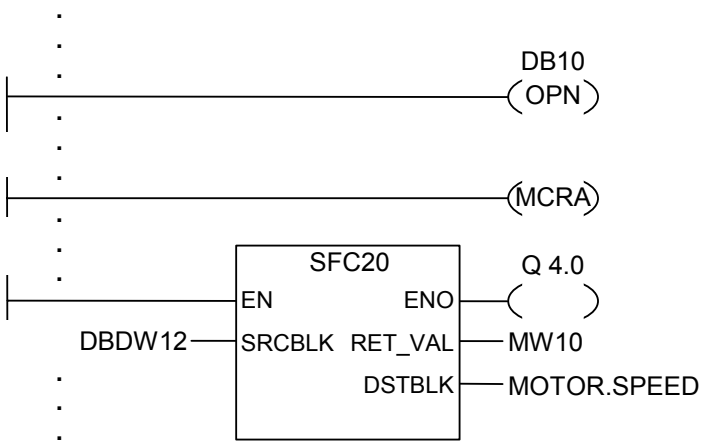
- Сохранение адреса возврата в вызывающий блок,
- Сохранение значений регистров блоков данных (DB и DI),
- Бит MA (бит активности функции MCR) сохраняется в BSTACK.
- Создание новой области в локальном стеке для вызываемой системной функции .

После этого выполняется программа вызываемой SFC. Параметр ENO = 1 если блок был обработан и не произошло ошибок .

Биты слова состояния

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Условный вызов	Записывает	X	-	-	-	0	0	X	X	X
Безусловный вызов	Записывает	-	-	-	-	0	0	X	X	X

Пример



Сегменты на рисунке являются частью пользовательского функционального блока в контактном плане . В этом блоке открывается блок DB10 и активируется функция MCR. Если выполняется безусловный вызов SFC20, происходит следующее:

Сохраняются точка возврата в вызывающий блок и регистры блоков данных для DB10 и экземплярного блока вызывающего FB. Бит MA , установленный в 1 инструкцией MCRA сохраняется в стеке блоков BSTACK и сбрасывается в 0 для вызванного блока SFC20. Выполнение программы продолжается в SFC20. После завершения обработки SFC20 , программа возвращается в вызывающий блок. Восстанавливается бит MA .

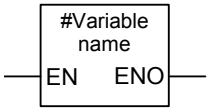
Если SFC20 обработан без ошибок, продолжается обработка в вызывающем блоке и состояние сигнала на выходе ENO = 1 и, соответственно, выход Q4.0 также равен 1. При ENO = 0, соответственно, и выход Q4.0 также равен 0.

Замечание

После возврата программы в вызывающий блок, не всегда гарантируется , что номер ранее открытого блока данных будет восстановлен. Для справки обратитесь к файлу README.

10.7 Вызов мультиэкземпляров

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
ENO	BOOL	I, Q, M, D, L,	Деблокировка выхода
# Variable name	FB/SFB	-	Имя мультиэкземпляра

Описание

Вы создаете мультиэкземпляр при помощи задания в таблице описаний функционального блока статической переменной типа FB. Только ранее описанные мультиэкземпляры будут отображаться в каталоге программных элементов. Интерфейс мультиэкземпляра зависит от описанных в нем переменных. Вход EN и выход ENO являются обязательными.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	0	0	X	X	X

10.8 Вызов библиотечных блоков

Библиотеки, установленные в SIMATIC Manager могут использоваться для вызова блоков, которые:

- Встроены в операционную систему Вашего CPU (Библиотека"Standard Library" в STEP 7 с версии 3 и "stdlibs (V2)" для STEP 7 версии 2)
- Вы создаете самостоятельно для последующего использования.

10.9 Важные замечания по использованию MCR функций

Будьте внимательны с блоками, в которых функция Master Control Relay активируется с помощью MCRA

- При деактивации MCR , значение 0 записывается всеми инструкциями присвоения в сегментах между Master Control Relay Вкл. и Master Control Relay Выкл. Это относится **ко всем** графическим элементам которые содержат присвоение, включая передачу параметров в блок.
- Функция MCR деактивируется при RLO = 0 перед инструкцией Master Control Relay Вкл .

Внимание: PLC в режиме STOP или в неопределенном состоянии!

Компилятор использует доступ на чтение к локальным данным , описанным как временные переменные VAR_TEMP для вычисления адреса. Это означает что следующие команды могут перевести PLC в **STOP** или **привести к неопределенному рабочему состоянию:**

Доступ к формальным параметрам

- Доступ к параметрам FC сложного типа STRUCT, UDT, ARRAY, STRING
- Доступ к параметрам FB сложного типа : STRUCT, UDT, ARRAY, STRING из области IN_OUT в блоке версии 2.
- Доступ к формальным параметрам функционального блока версии 2 если его адрес больше чем 8180.0.
- Доступ в функциональном блоке версии 2 к формальным параметрам типа BLOCK_DB, открывающим DB0. Некоторые обращения на доступ к данным , также могут перевести CPU в STOP, например: T 0, C 0, FC0, или FB0 для параметров типа TIMER, COUNTER, BLOCK_FC, и BLOCK_FB.

Передача параметров

- Вызов с передачей актуальных параметров.

LAD/FBD

- Т образные ветвления и коннекторы в КОР или FBD языках начинаются с RLO = 0.

Способ

Устранение зависимости нижележащих инструкций от MCR:

1. **Деактивируйте** Master Control Relay с использованием функции Master Control Relay Deactivate (деактивации главного управляющего реле) **до** опроса состояния в сегменте.
2. **Активируйте** Master Control Relay **снова** , используя инструкцию Master Control Relay Activate (активации главного управляющего реле) после выполнения критичных инструкций.

10.10 ---(MCR<) Включение Master Control Relay

Важные замечания по использованию MCR функций

Обозначение

---(MCR<)

Description

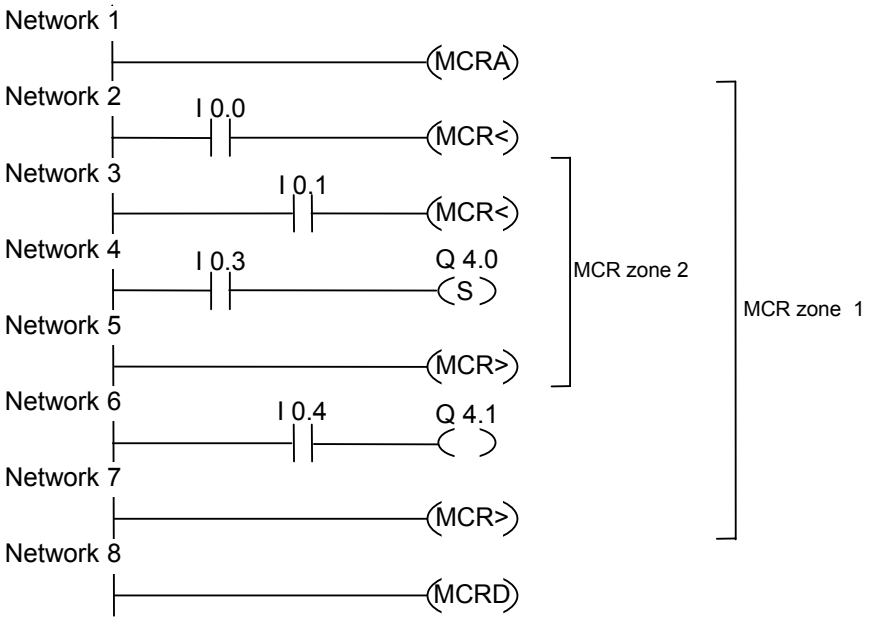
---(MCR<) Инструкция **Включить Master Control Relay (Главное управляющее реле)** запускает операцию, которая сохраняет RLO в стеке MCR и открывает зону действия MCR. Стек MCR работает по принципу буфера LIFO (Last In, First Out - последний вошел, первый вышел) и он имеет глубину в восемь записей. . Если стек уже полон, инструкция **---(MCR<)** генерирует ошибку стека MCR (MCRF). На следующие инструкции, описанные как MCR инструкции, оказывает влияние RLO, сохраненное в стеке MCR, при открытии зоны действия MCR:

- --(#) коннектор
- --( ) выход
- --(S) установить выход
- --(R) сбросить выход
- RS RS триггер
- SR SR триггер
- MOVE передача значения

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	0	1	-	0

Пример



Зона MCR активируется при выполнении инструкции MCRA. После этого возможно активировать до восьми вложенных зон MCR. В примере выше показаны две такие зоны. Функции выполняются следующим образом:

I0.0 = "1" (MCR зона 1 ВКЛ.): статус I0.4 передается на выход Q4.1

I0.0 = "0" (MCR зона 1 ВЫКЛ): Q4.1 в "0" независимо от статуса I0.4

I0.1 = "1" (MCR зона 2 ВКЛ): Q4.0 устанавливается в "1" если I0.3 в "1"

I0.1 = "0" (MCR зона 2 ВЫКЛ): Q4.0 остается неизменным независимо от статуса входа I0.3.

10.11 ---(MCR>) Выключение Master Control Relay

Важные замечания по использованию MCR функций

Обозначение

---(MCR>)

Описание

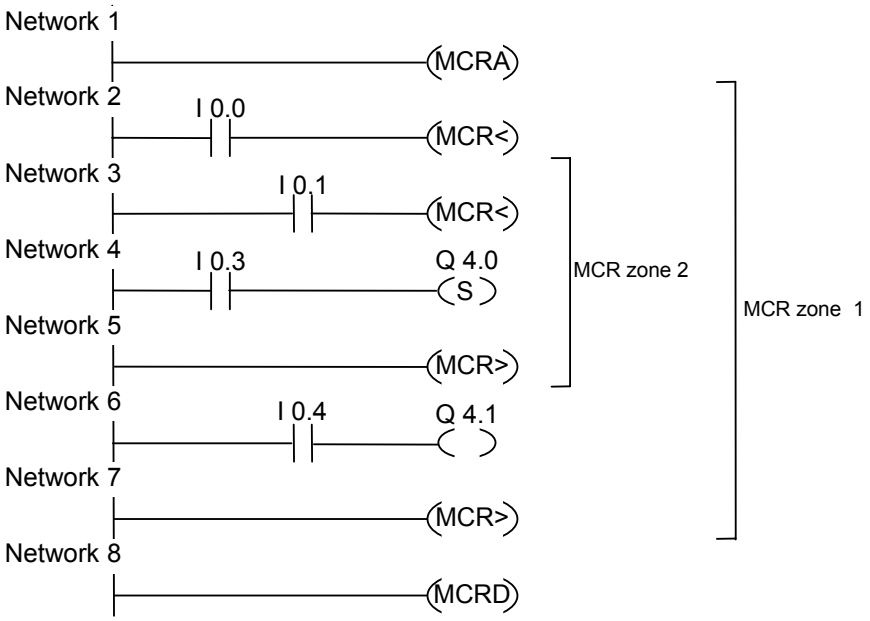
---(MCR>) Инструкция **Выключить Master Control Relay (Главное управляющее реле)** запускает операцию, которая восстанавливает RLO из стека MCR и закрывает зону действия MCR. Стек MCR работает по принципу буфера LIFO (Last In, First Out - последний вошел, первый вышел) и он имеет глубину в восемь записей. . Если стек уже пуст, инструкция **---(MCR>)** генерирует ошибку стека MCR (MCRF). На следующие инструкции, описанные как MCR инструкции, оказывает влияние RLO, сохраненное в стеке MCR, при открытии зоны действия MCR:

- --(#) коннектор
- --( ) выход
- --(S) установить выход
- --(R) сбросить выход
- RS RS триггер
- SR SR триггер
- MOVE передача значения

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	0	1	-	0

Пример



Зона MCR активируется при выполнении инструкции MCRA. После этого возможно активировать до восьми вложенных зон MCR. В примере выше показаны две такие зоны.

Первая инструкция ---(MCR>) (MCR ВЫКЛ) относится ко второй инструкции ---(MCR<) (MCR ВКЛ). Все инструкции между ними относятся ко 2 зоне MCR. Функции выполняются следующим образом:

I0.0 = "1: статус I0.4 передается на выход Q4.1

I0.0 = "0": Q4.1 в "0" независимо от статуса I0.4

I0.1 = "1: Q4.0 устанавливается в "1" если I0.3 в "1"

I0.1 = "0" :Q4.0 остается неизменным независимо от статуса входа I0.3.

10.12 ---(MCRA) Активация Master Control Relay

Важные замечания по использованию MCR функций

Обозначение

---(MCRA)

Описание

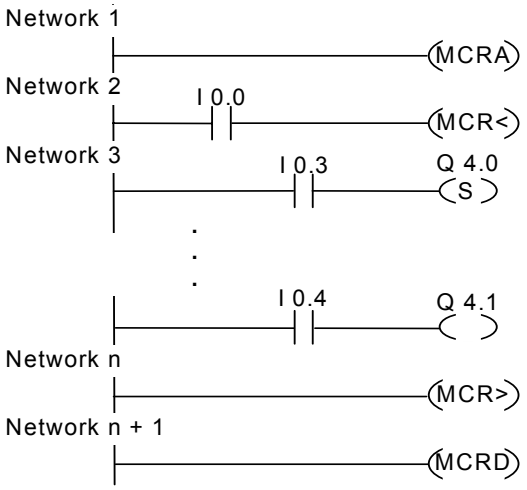
---(MCRA) (Активировать Master Control Relay) Эта инструкция активирует функцию главного управляющего реле(MCR). После выполнения этой инструкции можно программировать работу в зоне MCR с помощью инструкций:

- ---(MCR<)
- ---(MCR>)

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	-	-	-	-

Пример



MCR зона активируется с помощью инструкции MCRA. Инструкции между MCR< и MCR> (выходы Q4.0, Q4.1) обрабатываются следующим образом:

I0.0 = "1" (MCR ВКЛ): Q4.0 устанавливается в"1" если I0.3 имеет статус "1", останется неизменной при I0.3 в "0" и статус I0.4 присваивается выходу Q4.1.

I0.0 = "0" (MCR ВЫКЛ): Q4.0 остается неизменным независимо от статуса I0.3 и выход Q4.1 в "0" независимо от статуса входа I0.4

Инструкция ---(MCRD) деактивирует зону MCR. Это означает, что Вы не можете более использовать инструкции ---(MCR<) и ---(MCR>).

10.13 ---(MCRD) Деактивация Master Control Relay

Важные замечания по использованию MCR функций

Обозначение

---(MCRD)

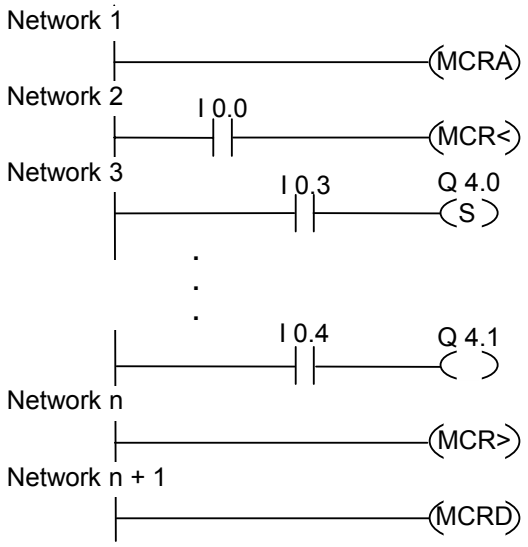
Описание

---(MCRD) (Деактивировать Master Control Relay) Эта инструкция деактивирует функцию главного управляющего реле(MCR). После выполнения этой инструкции нельзя программировать работу в зоне MCR.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	-	-	-	-	-	-	-	-	-

Пример



MCR зона активируется с помощью инструкции MCRA. Инструкции между MCR< и MCR> (выходы Q4.0, Q4.1) обрабатываются следующим образом:

I0.0 = "1" (MCR ВКЛ): Q4.0 устанавливается в"1" если I0.3 имеет статус "1", останется неизменной при I0.3 в "0" и статус I0.4 присваивается выходу Q4.1.

I0.0 = "0" (MCR ВЫКЛ): Q4.0 остается неизменным независимо от статуса I0.3 и выход Q4.1 в "0" независимо от статуса входа I0.4

Инструкция ---(MCRD) деактивирует зону MCR. Это означает, что вы не можете более использовать инструкции ---(MCR<) и ---(MCR>).

10.14 ---(RET) Возврат

Обозначение

---(RET)

Описание

Инструкция RET (Возврат) используется для условного окончания блока. Для этого выхода требуется выполнение некоторой предварительной инструкции.

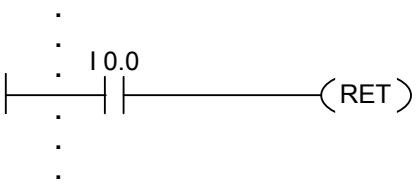
Биты слова состояния

Условный возврат (при RLO = "1"):

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает	*	-	-	-	0	0	1	1	0

* Инструкция **RET** стоит в одном ряду с такими инструкциями, как "SAVE; BEC, ". Она изменяет бит BR .

Пример



Обработка блока завершается при статусе I0.0 = "1".

11 Инструкции сдвига и циклического сдвига

11.1 Инструкции сдвига

11.1.1 Обзор инструкций сдвига

Описание

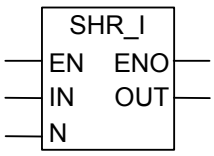
С помощью инструкций сдвига Вы можете побитно сдвигать содержимое входа IN (см. Регистры CPU) влево или вправо. Сдвиг на n битов влево умножает содержимое входа IN на 2^n; сдвиг на n битов вправо делит содержимое входа IN на 2^n. Например, если Вы сдвигаете значение 3 в двоичном коде на 3 бита влево, то получается десятичное значение 24. Если Вы сдвигаете десятичное значение 16 на 2 бита вправо, то в аккумуляторе получается двоичный эквивалент десятичного значения 4.

Число, задаваемое Вами для входного параметра N, показывает, на сколько битов должен производиться сдвиг. Разряды, освобождающиеся вследствие операции сдвига, заполняются нулями **или** состоянием знакового бита (”0” в случае положительного числа, ”1” в случае отрицательного числа). Бит, сдвигаемый последним, загружается в бит CC1 слова состояния . Биты CC0 и OV сбрасываются в “0”. Вы можете оценить бит CC1 слова состояния с помощью операций перехода. Вы можете использовать следующие инструкции сдвига:

- SHR_I : Сдвиг вправо числа типа Integer
- SHR_DI : Сдвиг вправо числа типа Double Integer
- SHL_W : Сдвиг слова влево
- SHR_W : Сдвиг слова вправо
- SHL_DW : Сдвиг двойного слова влево
- SHR_DW : Сдвиг двойного слова вправо

11.1.2 SHR_I : Сдвиг вправо числа Integer

Обозначение

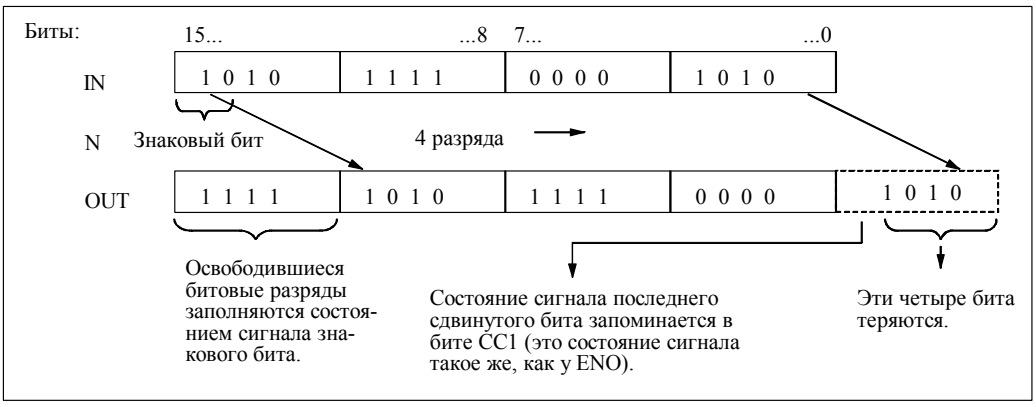


Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D, T, C	Разрешающий вход
ENO	BOOL	I, Q, M, L, D	Разрешающий выход
IN	INT	I, Q, M, L, D	Сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится сдвиг
OUT	INT	I, Q, M, L, D	Результат операции сдвиг

Описание

Инструкция **SHR_I** (**Сдвиг вправо целого числа)** активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно сдвигает вправо биты входа IN, имеющие номера с 0 по 15. Биты с 16 по 31 при этом не затрагиваются. Вход N задает на сколько бит происходит сдвиг. Если N больше, чем 16, то команда работает так, как будто N = 16. Битовые позиции слева заполняются состоянием сигнала бита 15 (разряд знака целого числа), то есть нулем, если число положительное, и 1, если число отрицательное. Результат операции сдвига Вы можете опрашивать на выходе OUT**.**

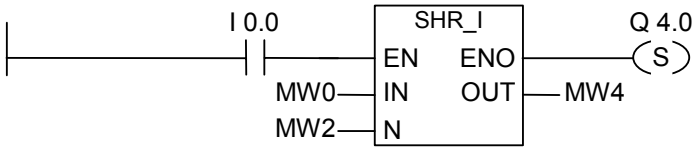
Инструкция, выполненная при неравном нулю N, всегда сбрасывает биты CC 0 и OV слова состояния в ”0”. Выход ENO повторяет состояние сигнала на входе EN.



Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	X	X	X	X	-	X	X	X	1

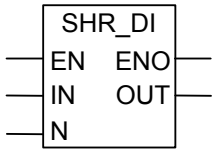
Пример



Операция активируется, если вход I0.0 равен 1. Меркерное слово MW0 сдвигается вправо на количество битов, заданное в MW2. Результат записывается в MW4, выход Q4.0 устанавливается в 1.
.

11.1.3 SHR_DI : Сдвиг вправо числа Double Integer

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D, T, C	Разрешающий вход
ENO	BOOL	I, Q, M, L, D	Разрешающий выход
IN	DINT	I, Q, M, L, D	Сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится сдвиг
OUT	DINT	I, Q, M, L, D	Результат операции сдвиг

Описание

Инструкция **SHR_DI** (**Сдвиг вправо числа Double Integer**) активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно сдвигает биты с 0 по 31вправо содержимого входа IN. Вход N задает, на сколько битов происходит сдвиг. Если N больше, чем 32, то команда работает так, как будто N = 32. Битовые позиции слева заполняются состоянием сигнала бита 31 (разряд знака двойного целого числа), то есть нулем, если число положительное, и 1, если число отрицательное. Результат операции сдвига Вы можете считывать на выходе OUT.

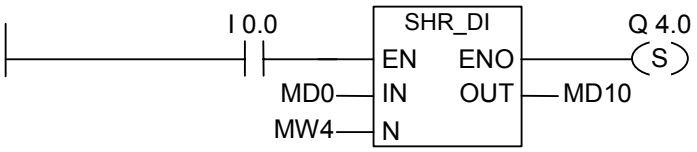
Операция, запущенная при не равном нулю N, всегда сбрасывает биты A0 и OV слова состояния в ”0”.

ENO повторяет состояние сигнала на входе EN.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
Записывает	X	X	X	X	-	X	X	X	1

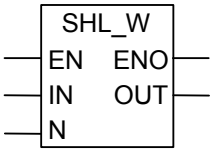
Пример



Операция активируется, если статус входа I0.0 равен 1. Двойное меркерное слово MD0 сдвигается вправо на количество бит, заданное MW4.
Результат сохраняется в MD10. Выход Q4.0 устанавливается в 1.

11.1.4 SHL_W : Сдвиг слова влево

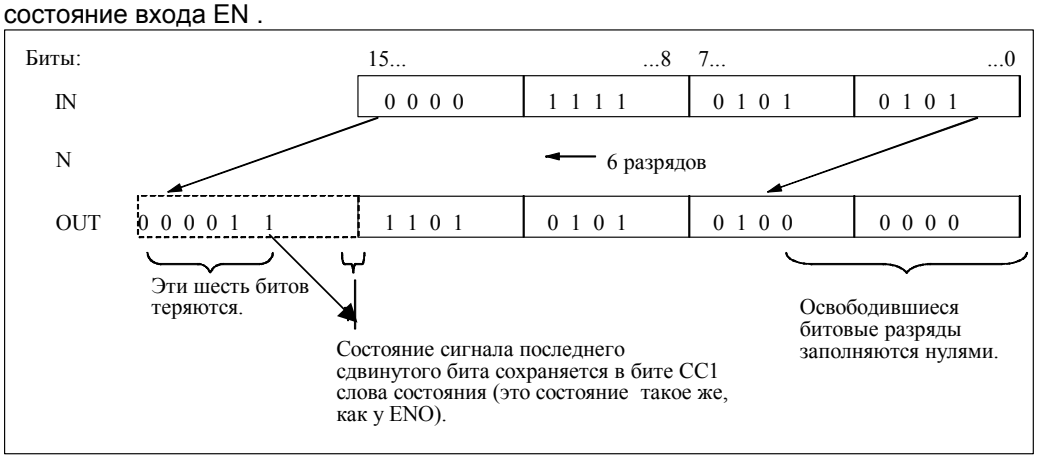
Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D, T, C	Разрешающий вход
ENO	BOOL	I, Q, M, L, D	Разрешающий выход
IN	WORD	I, Q, M, L, D	Сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится сдвиг
OUT	WORD	I, Q, M, L, D	Результат операции сдвиг

Описание

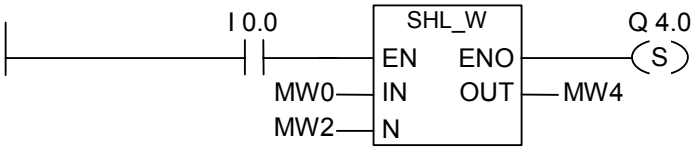
SHL_W: Инструкция **Сдвиг слова влево** активируется , если на разрешающем входе (EN)состояние сигнала =1 и побитно сдвигает влево биты с 0 по 15 входа IN. Вход N задает, на сколько бит происходит сдвиг. Если N больше, чем 16, то команда записывает 0 на выходе OUT и сбрасывает биты CC0 и OV слова состояния в ”0”. Освобождающиеся справа битовые позиции заполняются нулями. Результат операции сдвига может считываться на выходе OUT. Операция, запущенная при не равном нулю N, сбрасывает биты CC0 и OV слова состояния в ”0”. Выход ENO повторяет состояние входа EN .



Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает:	X	X	X	X	-	X	X	X	1

Пример



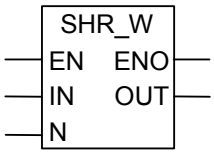
Операция активируется, если вход I0.0 равен 1.

Меркерное слово MW0 сдвигается влево на количество битов, заданное в MW2.

Результат сохраняется в MW4. Выход Q4.0 устанавливается в 1.

11.1.5 SHR_W: Сдвиг слова вправо

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D, T, C	Разрешающий вход
ENO	BOOL	I, Q, M, L, D	Разрешающий выход
IN	WORD	I, Q, M, L, D	Сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится сдвиг
OUT	WORD	I, Q, M, L, D	Результат операции сдвиг

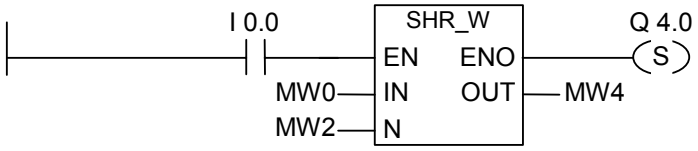
Описание

SHR_W :Инструкция **сдвиг слова вправо** активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно сдвигает вправо биты входа IN, имеющие номера с 0 по 15. Биты с номерами с 16 по 31 не изменяются. Вход N задает, на сколько бит происходит сдвиг. Если N больше, чем 16, то команда записывает 0 на выходе OUT и сбрасывает биты CC0 и OV в ”0”. Освобождающиеся слева битовые позиции заполняются нулями. Результат операции сдвига Вы можете опрашивать на выходе OUT. Операция, запущенная при не равном нулю N, всегда сбрасывает биты CC0 и OV слова состояния в ”0”. Выход ENO имеет то же значение , что и EN.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	X	X	X	X	-	X	X	X	1

Пример

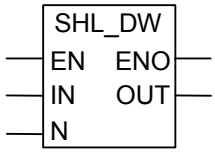


Инструкция активируется, если вход I0.0 равен 1. Меркерное слово MW0 сдвигается вправо на количество бит, заданное в MW2. Результат сохраняется в MW4 . Выход Q4.0 устанавливается в 1.

.

11.1.6 SHL_DW: Сдвиг двойного слова влево

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D, T, C	Разрешающий вход
ENO	BOOL	I, Q, M, L, D	Разрешающий выход
IN	DWORD	I, Q, M, L, D	Сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится сдвиг
OUT	DWORD	I, Q, M, L, D	Результат операции сдвиг

Описание

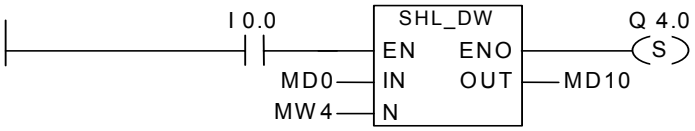
SHL_DW Инструкция **Сдвиг двойного слова влево** активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно сдвигает влево биты входа IN, имеющие номера с 0 по 31. Вход N задает, на сколько бит происходит сдвиг. Если N больше, чем 32, то команда записывает 0 на выходе OUT и сбрасывает биты CC0 и OV слова состояния в ”0”. Освобождающиеся справа битовые позиции заполняются нулями. Результат операции сдвига Вы можете считывать на выходе OUT.

Инструкция, запущенная при не равном нулю N, всегда сбрасывает биты CC0 и OV слова состояния в ”0”. Выход ENO всегда имеет то же значение , что и вход EN.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
Записывает	x	x	x	x	-	x	x	x	1

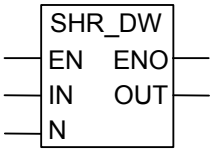
Пример



Инструкция активируется, если I0.0 равно1. Двойное меркерное слово MD0 смещается влево на количество бит, заданное в MW4. Результат сохраняется в MD10. Выход Q4.0 устанавливается, если последний сдвинутый бит имеет состояние сигнала 1.

11.1.7 SHR_DW : Сдвиг двойного слова вправо

Обозначение

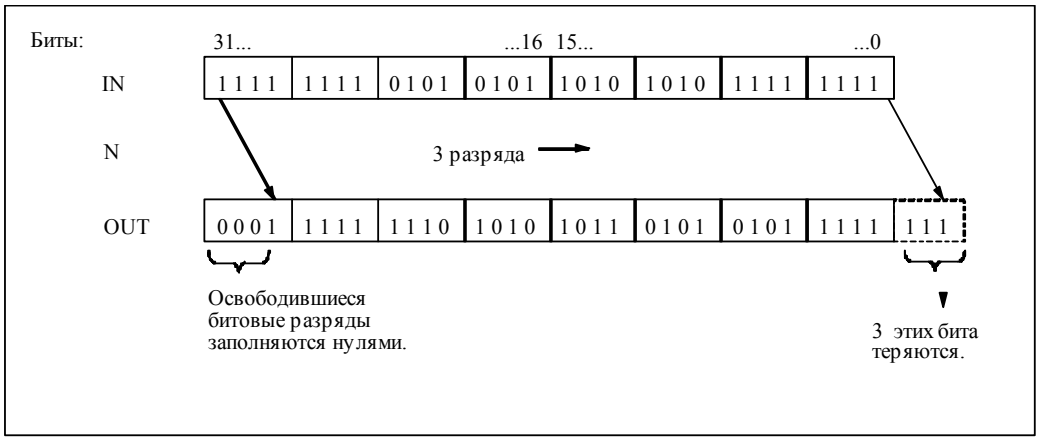


Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Разрешающий вход
ENO	BOOL	I, Q, M, L, D	Разрешающий выход
IN	DWORD	I, Q, M, L, D	Сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится сдвиг
OUT	DWORD	I, Q, M, L, D	Результат операции сдвига

Описание

SHR_DW: Инструкция **Сдвиг двойного слова вправо** активируется состоянием сигнала “1”на разрешающем входе (EN) и побитно сдвигает вправо биты входа IN, имеющие номера с 0 по 31. Вход N задает, на сколько бит происходит сдвиг. Если N больше, чем 32, то команда записывает 0 на выходе OUT и сбрасывает биты CC0 и OV в ”0”. Освобождающиеся слева битовые позиции заполняются нулями. Результат операции сдвига Вы можете опрашивать на выходе OUT.

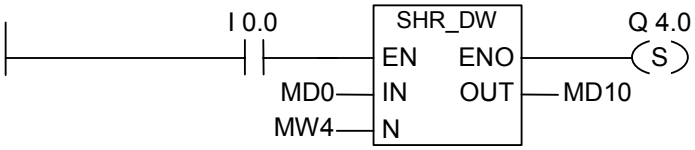
Инструкция, запущенная при не равном нулю N, всегда сбрасывает биты CC0 и OV слова состояния в ”0”. Выход ENO равен входу EN.



Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	X	X	X	X	-	X	X	X	1

Пример



Операция активируется, если I0.0 = 1. Меркерное слово MD0 сдвигается вправо на количество бит, заданное в MW4.
Результат сохраняется в MD10. Выход Q4.0 устанавливается в 1.

11.2 Инструкции циклического сдвига

11.2.1 Обзор инструкций циклического сдвига

Описание

С помощью инструкций циклического сдвига, Вы можете побитно циклически сдвигать вправо или влево все содержимое входа IN. Освобождающиеся разряды заполняются состояниями сигналов тех битов, которые выталкиваются из входного значения IN. Число, задаваемое Вами для входного параметра N, показывает, на сколько битов должен производиться циклический сдвиг.

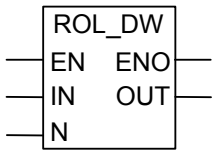
В зависимости от выбранной операции, циклический сдвиг происходит через бит CC1. Бит CC0 слова состояния сбрасывается в “0”.

В Вашем распоряжении имеются следующие операции циклического сдвига:

- ROL_DW : Циклический сдвиг двойного слова влево
- ROR_DW : Циклический сдвиг двойного слова вправо

11.2.2 ROL_DW : Циклический сдвиг двойного слова влево

Обозначение

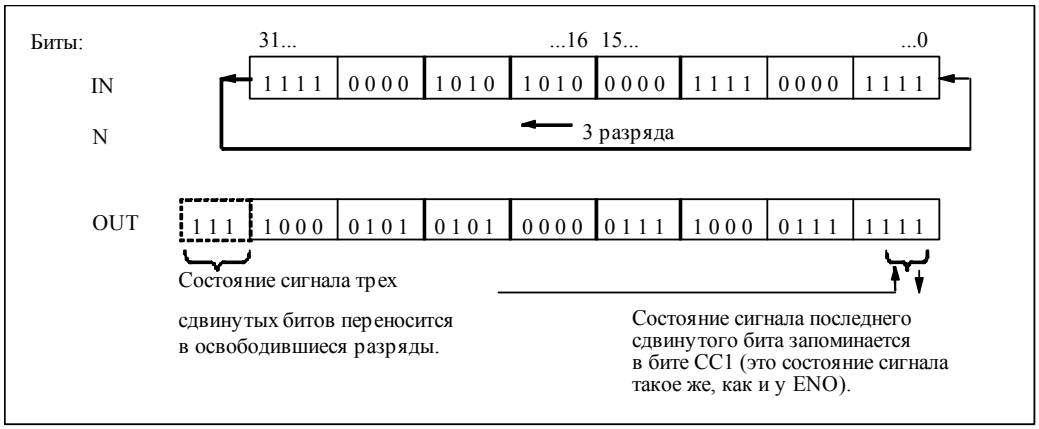


Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Разрешающий вход
ENO	BOOL	I, Q, M, L, D	Разрешающий выход
IN	DWORD	I, Q, M, L, D	Циклически сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится цикл. сдвиг
OUT	DWORD	I, Q, M, L, D	Результат циклического сдвига

Описание

ROL_DW Инструкция **Циклический сдвиг двойного слова влево** активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно циклически сдвигает влево все содержимое входа IN. Вход N задает, на сколько бит происходит циклический сдвиг. Если N больше, чем 32, то двойное слово циклически сдвигается на число бит, равное (((N-1) по модулю 32) +1). Освобождающиеся справа битовые позиции заполняются состояниями сигналов циклически сдвигаемых битов. Результат операции циклического сдвига Вы можете опрашивать на выходе OUT.
Инструкция, запущенная при не равном нулю N, всегда сбрасывает биты CC0 и OV слова состояния в ”0”. Выход ENO всегда равен входу EN.

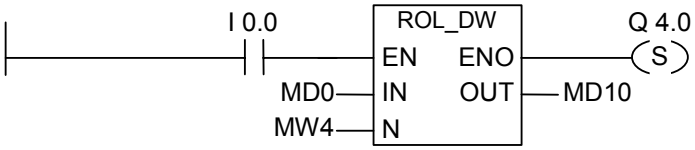
.



Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	X	X	X	X	-	X	X	X	1

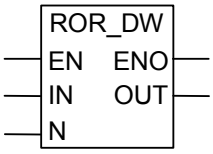
Пример



Операция активируется, если I0.0 = 1.Двойное меркерное слово MD0 сдвигается циклически влево на количество бит, заданное в MW4. Результат сохраняется в MD10. Выход Q4.0 устанавливается в 1.

11.2.3 ROR_DW : Циклический сдвиг двойного слова вправо

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Разрешающий вход
ENO	BOOL	I, Q, M, L, D	Разрешающий выход
IN	DWORD	I, Q, M, L, D	Циклически сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится цикл, сдвиг
OUT	DWORD	I, Q, M, L, D	Результат циклического сдвига

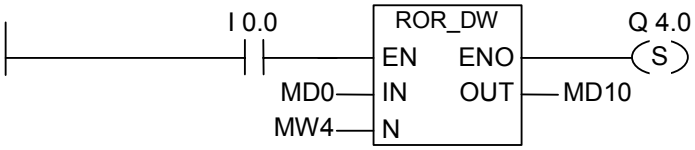
Описание

ROR_DW Операция **Циклический сдвиг двойного слова вправо** активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно задает, на сколько битов происходит циклический сдвиг. Если N больше, чем 32, то двойное слово циклически сдвигается на число битов, равное (((N-1) по модулю 32) +1). Значение N может находиться между 0 и 31. Освобождающиеся слева битовые позиции заполняются состояниями сигналов циклически сдвигаемых битов. Результат операции циклического сдвига Вы можете опрашивать на выходе OUT.
Операция, запущенная при не равном нулю N, всегда сбрасывает биты CC0 и OV слова состояния в ”0”. Выход ENO всегда имеет то же значение , что и EN.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	X	X	X	X	-	X	X	X	1

Пример



Операция активируется, если I0.0 = 1.Двойное меркерное слово MD0 циклически сдвигается вправо на количество битов, заданное в MW4.

Результат сохраняется в MD10. Выход Q4.0 устанавливается в 1.

12 Инструкции с битами слова состояния

12.1 Обзор инструкций с битами слова состояния

Описание

Инструкции с битами слова состояния являются битовыми логическими инструкциями, которые работают с битами одного из регистров CPU : “Словом состояния”. Каждая из этих инструкций реагирует на одно из следующих условий, отображаемых одним или несколькими битами слова состояния:

- Бит двоичного результата (BR ---I I---) установлен в 1.
- Математическая операция привела к переполнению (OV ---I I---) , или переполнению с запоминанием (OS ---I I---).
- Результат арифметической операции недопустим (UO ---I I---).
- Результат математической операции находится по отношению к 0 в одном из следующих состояний: == 0, <> 0, > 0, < 0, >= 0, <= 0.

Когда биты состояния оценивают в последовательной логической цепочке, результат их опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для И . Когда биты состояния оценивают в параллельной логической цепочке, результат их опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для ИЛИ.

Биты слова состояния

Слово состояния представляет собой регистр памяти Вашего CPU, к которому Вы можете обращаться по адресу бита и в поразрядных логических операциях над словами. Структура слова состояния:

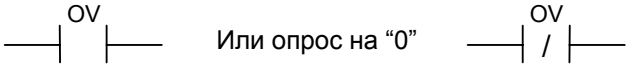
2^{15}... ...2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC

Вы можете оценивать биты слова состояния при работе с функциями:

- математических инструкций с целыми числами
- математических инструкций над числами с плавающей точкой

12.2 OV ---| |--- Бит ошибки “Переполнение”

Обозначение



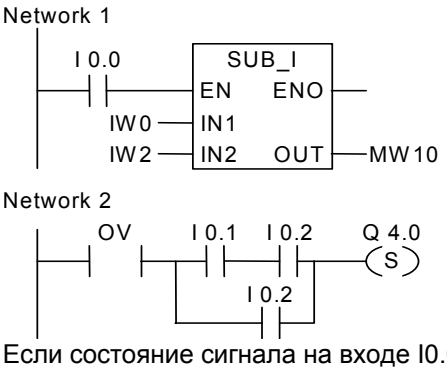
Описание

OV ---| |---: Вы можете использовать инструкцию опроса **Бита ошибки “Переполнение”** для обнаружения переполнения (OV) в последней математической операции. Если после выполнения математической операции результат оказывается за пределами допустимого диапазона в отрицательной или положительной области, то бит OV в слове состояния устанавливается в 1. При оценке результата опроса в последовательной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для И . При оценке результата опроса в параллельной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для ИЛИ.

Биты слова состояния

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

Пример



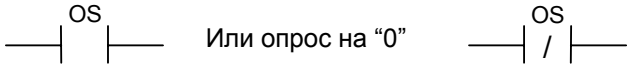
Если состояние сигнала на входе I0.0 равно 1, то блок SUB_I выполняется. Если результат арифметической операции вычитания входного слова IW2 из входного слова IW0 находится за пределами допустимого диапазона для целых чисел то бит OV в слове состояния устанавливается в 1. Результат опроса бита OV равен 1. Выход Q4.0 устанавливается, если опрос OV равен 1 и RLO сегмента 2 равен 1 (если RLO перед выходом Q4.0 равен 1).

Замечание

Опрос OV необходим поскольку здесь используются два разных сегмента. Другим вариантом является опрос выхода ENO который будет выдавать статус "0" если будет получено переполнение при выполнении инструкции.

12.3 OS ---| |--- : Бит ошибки “Переполнение с запоминанием”

Обозначение



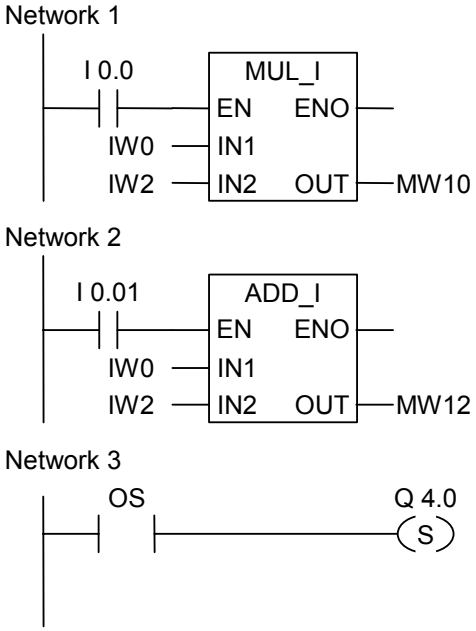
Описание

OS ---| |--- : Вы можете использовать команду опроса **Бит ошибки “Переполнение с запоминанием”** или **OS ---| / |---**(опрос на отсутствие переполнения) для распознавания предыдущего переполнения в математических операциях. Если после ее выполнения результат оказывается за пределами допустимого диапазона в отрицательной или положительной области, то бит OS в слове состояния устанавливается в 1. Команда опрашивает состояние этого бита. В отличие от бита OV (переполнение) бит OS остается установленным, даже если следующие арифметические операции были выполнены без ошибок. При оценке результата опроса в последовательной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для И . При оценке результата опроса в параллельной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для ИЛИ.

Биты слова состояния

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

Пример



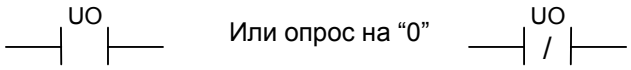
Если состояние сигнала на входе I0.0 равно 1, то активируется блок MUL_I. Если состояние сигнала на входе I0.1 равно 1, то активируется блок ADD_I. Если результат одной из этих арифметических операций выходит за пределы допустимого диапазона для целых чисел, то бит OS в слове состояния уста навливается. Результат опроса состояния бита OS в случае переполнения равен 1 и выход Q4.0 устанавливается в 1.

Замечание

Опрос бита OS необходим поскольку здесь используются два разных сегмента. Другим вариантом является опрос выходов ENO которые будут выдавать статус "0" если будет получено переполнение при выполнении инструкции и произвести их логическое сопряжение.

12.4 UO ---| |--- : Бит ошибки “Неупорядочено”

Обозначение



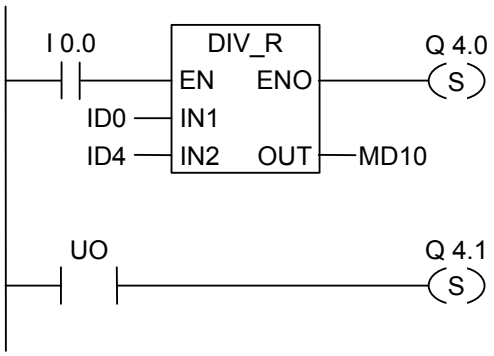
Описание

UO ---| |--- : или **UO ---| / |--- :** Вы можете использовать инструкцию опроса **Бита ошибки “Неупорядочено”** для проверки, является ли результат математической инструкции с плавающей точкой неупорядоченным (иными словами, не является ли одно из значений, с которым выполняется операция, недопустимым числом с плавающей точкой). Если результат математической операции является неупорядоченным (UO = unordered - неупорядочен), то опрос состояния сигнала дает результат 1. Если комбинация CC1 и CC0 не указывает на неупорядоченность результата математической операции, то опрос состояния сигнала дает 0.

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	X	X	X	1

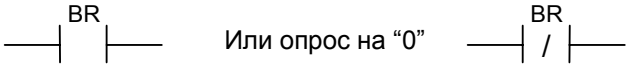
Пример



Если состояние сигнала на входе I0.0 равно 1, то блок DIV_R активизируется. Если значение одного из двойных входных слов ID0 или ID4 является недопустимым числом с плавающей точкой, то арифметическая операция с плавающей точкой неупорядочена. Если состояние сигнала EN равно 1 (активирован) и при выполнении команды возникает ошибка, то состояние сигнала на ENO равно 0. Выход Q4.1 устанавливается, если функция DIV_R выполняется, но одно из значений в арифметической функции является недопустимым числом с плавающей точкой.

12.5 BR ---| |--- : Бит ошибки “Двоичный результат”

Обозначение



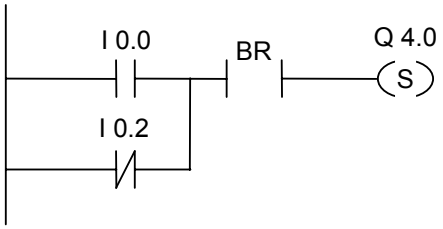
Описание

BR ---| |--- (опрос бита ошибки BR) или **BR ---| / |---** (опрос на «0» бита BR) используются для проверки логического состояния бита BR в слове состояния. При оценке результата опроса в последовательной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для И . При оценке результата опроса в параллельной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для ИЛИ.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
Записывает	-	-	-	-	-	X	X	X	1

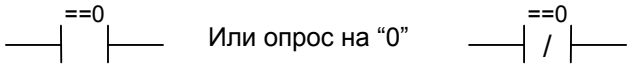
Пример



Выход Q4.0 устанавливается в “1” если статус сигнала на I0.0 = 1 ИЛИ статус сигнала на входе I0.2 = 0, И, в дополнение к этому результату, статус сигнала бита BR = 1.

12.6 ==0 ---| |--- Бит результата “Равно 0”

Обозначение



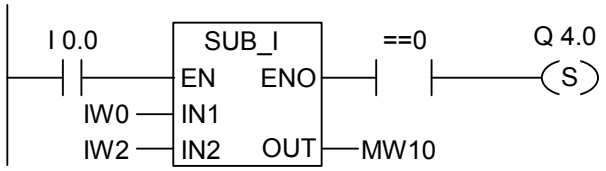
Описание

==0 ---| |--- (результат = 0) или**==0 ---| / |---** (результат не равен 0) используются для оценки результата математической операции на равенство "0". При выполнении этой инструкции оцениваются биты CC 1 и CC 0 в слове состояния для определения равенства результата "0". При оценке результата опроса в последовательной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для И . При оценке результата опроса в параллельной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для ИЛИ.

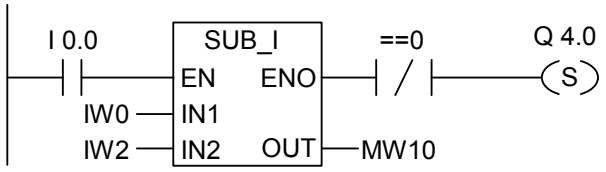
Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	X	X	X	1

Пример



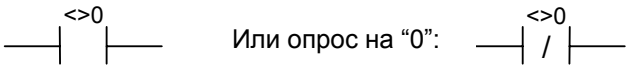
Если состояние сигнала на входе I0.0 равно 1, то блок SUB_I активируется. Если значение входного слова IW0 больше значения входного слова IW2, то результат операции IW0 - IW2 больше 0. Если состояние сигнала на EN равно 1 (активирован) и при выполнении операции получается нулевой результат, выход Q 4.0 устанавливается в 1.



Выход Q4.0 устанавливается, если функция выполняется правильно и результат не равен 0.

12.7 <>0 ---| |--- Бит результата “Не равно 0”

Обозначение



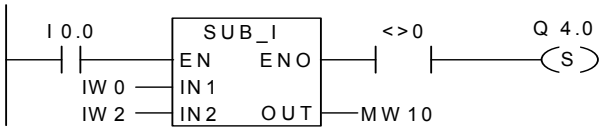
Описание

<>0 ---| |--- (Бит результата не равно 0) или **<>0 ---| / |---** (опрос на “0” бит а результата не равно 0) используются для оценки результата математической операции на неравенство "0". При выполнении этой инструкции оцениваются биты CC 1 и CC 0 в слове состояния для определения неравенства результата "0". При оценке результата опроса в последовательной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для И . При оценке результата опроса в параллельной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для ИЛИ.

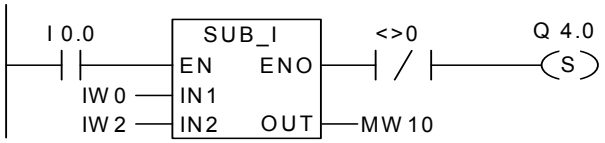
Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	x	x	x	1

Пример



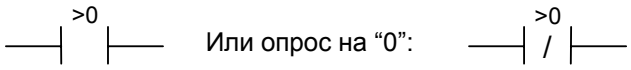
При I0.0 = 1, блок SUB_I активируется. Если значение входного слова IW0 не равно значению в IW2, то результат операции IW0 - IW2 не равен 0. Если состояние сигнала на EN равно 1 (активирован) и при выполнении операции получается ненулевой результат, выход Q 4.0 устанавливается в 1.



Выход Q4.0 устанавливается, если функция выполняется правильно и результат равен 0

12.8 >0 ---| |--- Бит результата “Больше 0”

Обозначение



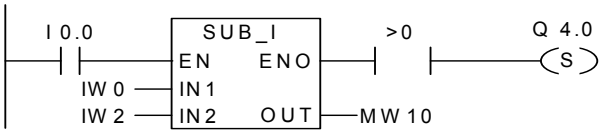
Описание

>0 ---| |--- (Бит результата “Больше 0”) или **>0 ---| / |---** (Опрос на “0” бита результата “Больше 0”) используются для оценки результата математической операции на больше "0". При выполнении этой инструкции оцениваются биты CC 1 и CC 0 в слове состояния для определения знака результата . При оценке результата опроса в последовательной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для И . При оценке результата опроса в параллельной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для ИЛИ.

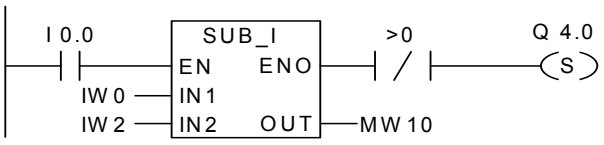
Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	x	x	x	1

Пример



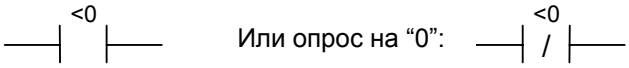
При I0.0 = 1, блок SUB_I активируется. Если значение входного слова IW0 больше значения в IW2, то результат операции IW0 - IW2 больше 0. Если состояние сигнала на EN равно 1 (активирован) и при выполнении операции получается положительный результат, выход Q 4.0 устанавливается в 1.



Выход Q4.0 устанавливается, если функция выполняется правильно и результат не больше 0.

12.9 <0 ---| |--- Бит результата “Меньше 0”

Обозначение



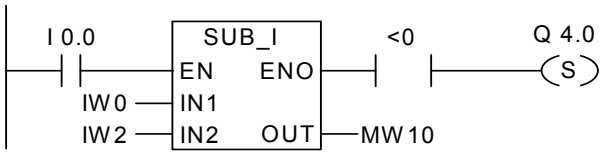
Описание

<0 ---| |--- (Бит результата ”Меньше 0”) или **<0 ---| / |---** (Опрос на “0” бита результата ”Меньше 0”) используются для оценки результата математической операции на меньше "0". При выполнении этой инструкции оцениваются биты CC 1 и CC 0 в слове состояния для определения знака результата . При оценке результата опроса в последовательной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для И . При оценке результата опроса в параллельной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для ИЛИ.

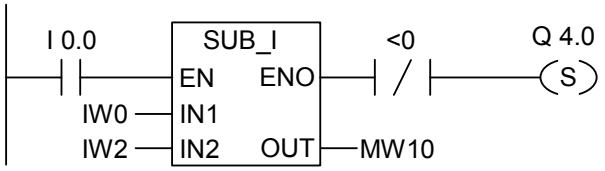
Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	x	x	x	1

Пример



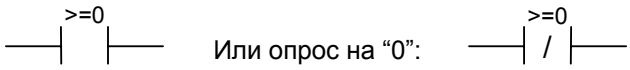
При I0.0 = 1, блок SUB_I активируется. Если значение входного слова IW0 меньше значения в IW2, то результат операции IW0 - IW2 меньше 0. Если состояние сигнала на EN равно 1 (активирован) и при выполнении операции получается отрицательный результат, выход Q 4.0 устанавливается в 1 .



Выход Q4.0 устанавливается, если функция выполняется правильно и результат не меньше 0.

12.10 >=0 ---| |--- Бит результата “Больше или равно 0”

Обозначение



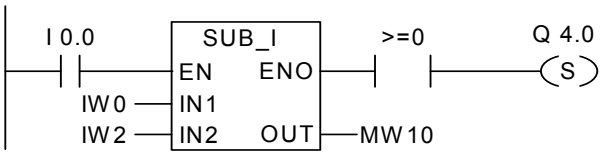
Описание

>=0 ---| |--- (Бит результата “Больше или равно 0”) или **>=0 ---| / |---** (Опрос на “0” бита результата “Больше или равно 0”) используются для оценки результата математической операции на больше или равно "0". При выполнении этой инструкции оцениваются биты CC 1 и CC 0 в слове состояния для определения знака результата . При оценке результата опроса в последовательной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для И . При оценке результата опроса в параллельной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для ИЛИ.

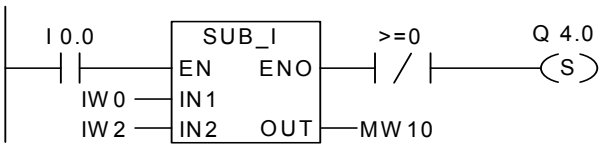
Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	x	x	x	1

Пример



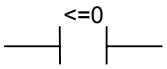
При I0.0 = 1, блок SUB_I активируется. Если значение входного слова IW0 больше или равно значению в IW2, то результат операции IW0 - IW2 больше или равен 0. Если состояние сигнала на EN равно 1 (активирован) и при выполнении операции получается неотрицательный результат, выход Q 4.0 = 1



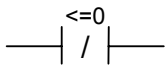
Выход Q4.0 устанавливается, если функция выполняется правильно и результат меньше 0.

12.11 <=0 ---| |--- Бит результата “Меньше или равно 0”

Обозначение



Или опрос на “0”:



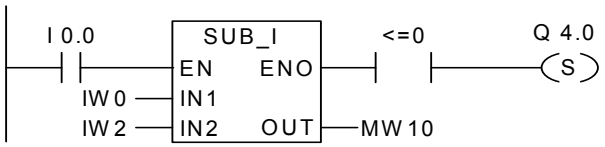
Описание

<0 ---| |--- (Бит результата ”Меньше или равно 0”) или **<0 ---| / |---** (Опрос на “0” бита результата ”Меньше или равно 0”) используются для оценки результата математической операции на меньше или равно "0". При выполнении этой инструкции оцениваются биты CC 1 и CC 0 в слове состояния для определения знака результата . При оценке результата опроса в последовательной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для И . При оценке результата опроса в параллельной логической цепочке, результат опроса сопрягается с предшествующим результатом логической операции в соответствии с таблицей истинности для ИЛИ.

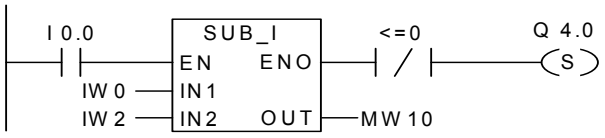
Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Записывает:	-	-	-	-	-	x	x	x	1

Примеры



При I0.0 = 1, блок SUB_I активируется. Если значение входного слова IW0 меньше или равен значению в IW2, то результат операции IW0 - IW2 меньше или равен 0. Если состояние сигнала на EN равно 1 и при выполнении операции получается число <=0, выход Q 4.0 устанавливается в 1 .



Выход Q4.0 устанавливается, если функция выполняется правильно и результат больше 0.

13 Таймерные инструкции

13.1 Обзор таймерных инструкций

Описание

Для правильного выбора таймера и задания значения времени дополнительную информацию Вы найдете в руководстве "Location of a Timer in Memory and Components of a Timer"(Расположение таймеров в памяти и компоненты таймера).

К области памяти таймеров имеют доступ следующие функции:

- S_PULSE : Задание параметров и запуск S5 таймера «Импульс»
- S_PEXT : Задание параметров и запуск S5 таймера«Удлиненный импульс»
- S_ODT : Задание параметров и запуск S5 таймера «Задержка включения»
- S_ODTS : Задание параметров и запуск S5 таймера «Задержка включения с памятью»
- S_OFFDT : Задание параметров и запуск S5 таймера «Задержка выключения»
- ---(SP) Катушка таймера «Импульс»
- ---(SE) Катушка таймера «Удлиненный импульс»
- ---(SD) Катушка таймера «Задержка включения»
- ---(SS) Катушка таймера «Задержка включения с памятью»
- ---(SF) Катушка таймера «Задержка выключения»

13.2 Область памяти и компоненты таймера

Область памяти

Таймеры имеют область, зарезервированную для них в памяти Вашего CPU. Эта область памяти резервирует одно 16-битное слово для каждого таймерного адреса. При программировании в КОР поддерживаются 256 таймеров. Для определения точного количества таймеров, Вам необходимо обратиться к руководству на Ваш контроллер. К области памяти таймеров имеют доступ следующие функции:

- Таймерные инструкции
- Актуализация таймерных слов генератором тактовых импульсов. В режиме RUN эта функция CPU уменьшает заданное значение времени на одну единицу с интервалом, установленным 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 временных отрезков. Когда таймер актуализируется, значение времени уменьшается на одну единицу через интервалы, установленные базой времени. Значение времени уменьшается до тех пор, пока оно не станет равным нулю. Вы можете задавать значение времени в двоичном, шестнадцатиричном или двоично-десятичном коде(BCD).

Вы можете загрузить значение времени с использованием следующего синтаксиса:

- W#16#wxyz , где

  -w - база времени (временной интервал или разрешение)

  -xyz – значение времени в BCD коде

- S5T#a**H**_b**M**_c**S**_d**MS**
  - где: a = часы, b = минуты, c = секунды и d = миллисекунды
  - База времени выбирается автоматически и значение округляется до ближайшего меньшего числа с этой базой времени.

Максимальное время, которое Вы можете ввести, составляет 9 990 секунд или 2H_46M_30S.

S5TIME#4S = 4 секунды
s5t#2h_15m = 2 часа и 15 минут
S5T#1H_12M_18S = 1 час, 12 минут и 18 секунд

База времени

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

База времени	Двоичный код для базы времени
10 ms	00
100 ms	01
1 s	10
10 s	11

Значения больше 2h46m30s не допускаются. Значения, разрешающая способность которых слишком велика для требуемого диапазона, например 2h10ms, округляются таким образом, что достигается требуемый диапазон, но не желаемая разрешающая способность. Следующая таблица показывает возможные разрешающие способности и соответствующие диапазоны:

Разрешающая способность	База времени
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 ячейке таймера

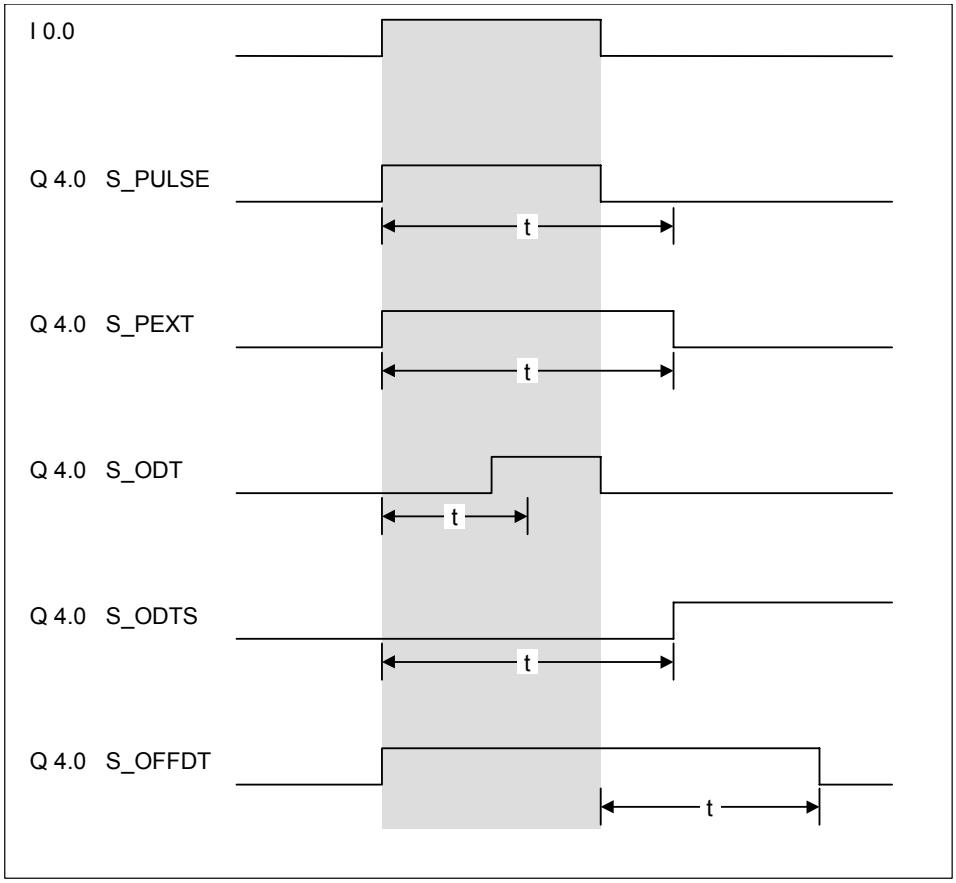
При запуске таймера, содержимое таймерной ячейки используется в качестве значения времени. Биты с 0 по 11 в таймерной ячейке содержат значение времени в двоично-десятичном формате (BCD-формат: каждая группа из четырех битов содержит двоичный код десятичного разряда). Биты 12 и 13 содержат базу времени в двоичном коде . Следующий рисунок показывает содержимое таймерной ячейки, загруженной значением времени 127 с базой времени 1 секунда:

Чтение времени и базы времени

Каждый таймерный блок предоставляет два выхода, BI и BCD, для которых Вы можете задать адрес слова. Выход BI предоставляет значение времени в двоичном формате, база времени не отображается. Выход BCD предоставляет базу времени и значение времени в двоично-десятичном формате (BCD).

Выбор подходящего таймера

Этот рисунок поможет Вам выбрать нужный тип таймера для обработки временных событий:

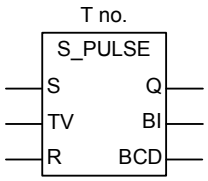


Таймер	Описание
S_PULSE Таймер «Импульс»	Максимальное время в течение которого выходной сигнал остается равным 1, совпадает с запрограммированным временем T. Выход сбрасывается раньше, если входной сигнал меняется состояние на 0.
S_PEXT Таймер «Импульс с памятью»	Выходной сигнал остается равным 1 в течение запрограммированного времени независимо от того, как долго остается равным 1 входной сигнал.
S_ODT Таймер «Задержка включения»	Выходной сигнал устанавливается в 1 только по истечении запрограммированного времени, при этом входной сигнал все еще должен быть равен 1.
S_ODTS Таймер «Задержка включения с памятью»	Выходной сигнал устанавливается в 1 только по истечении запрограммированного времени независимо от того, как долго остается равным 1 входной сигнал.
S_OFFDT Таймер «Задержка выключения»	Выходной сигнал устанавливается в 1, когда устанавливается в 1 входной сигнал, и остается равным 1, пока таймер работает. Отсчет времени начинается, когда входной сигнал меняется с 1 на 0.

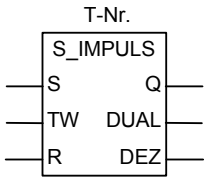
13.3 S_PULSE : Задание параметров и запуск таймера «Импульс»

Обозначение

Английский



Немецкий



Параметр Английский	Параметр Немецкий	Тип данных	Область памяти	Описание
T no.	T Nr.	TIMER	T	Номер таймера. Диапазон номеров зависит от CPU.
S	S	BOOL	I, Q, M, D, L	Вход запуска
TV	TW	S5TIME	I, Q, M, D, L	Установка времени(от 0-9990)
R	R	BOOL	I, Q, M, D, L	Вход сброса
BI	DUAL	WORD	I, Q, M, D, L	Остаток времени (значение в двоичном коде)
BCD	DEZ	WORD	I, Q, M, D, L	Остаток времени (значение в формате BCD)
Q	Q	BOOL	I, Q, M, D, L	Состояние таймера

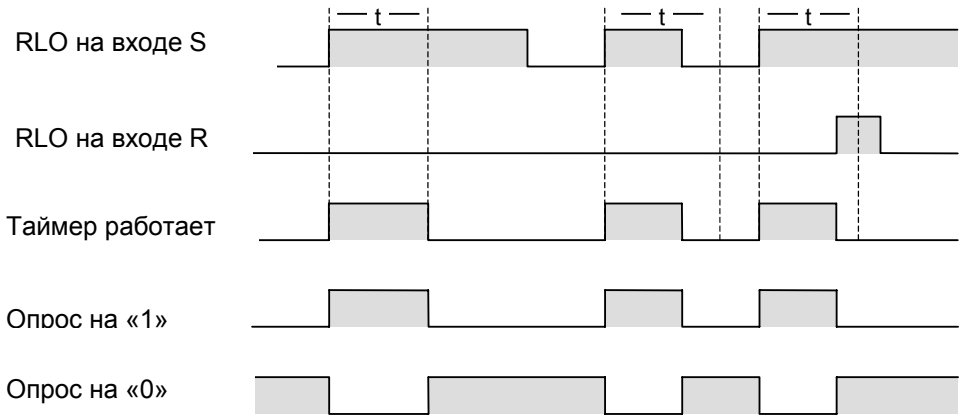
Описание

S_PULSE : (S5 таймер «Импульс») запускает заданный таймер по нарастающему фронту (изменение состояния сигнала с 0 на 1) на входе запуска (S). Для запуска таймера всегда необходимо изменение сигнала. Таймер продолжает работать в течение времени, заданного на входе TV, пока состояние сигнала на входе S остается равным 1. Пока таймер работает, опрос выхода Q на высокий уровень дает результат логической операции 1. Если на входе S сигнал меняется с 1 на 0 до истечения заданного времени, таймер останавливается. Тогда опрос состояния сигнала на 1 на выходе Q дает 0. Если во время работы таймера происходит изменение с 0 на 1 сигнала на входе сброса (R), то таймер сбрасывается. Это изменение сбрасывает в ноль время и базу времени. Единица на входе R таймера не оказывает никакого влияния если таймер не работает.

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на выходе BI представлено в двоичном формате, а на BCD - в двоично-десятичном формате. Текущее время равно разнице между начальным значением, заданным на входе TV и временем , прошедшим с момента запуска таймера.

Временные диаграммы

Импульсный таймер:

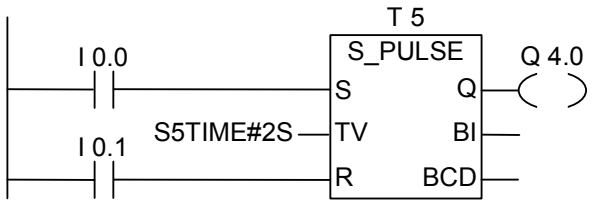


T = Заданное время

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	X	X	X	1

Пример

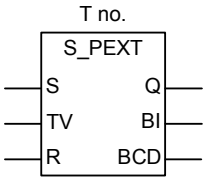


Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт RLO), таймер T5 запускается. Таймер продолжает работать с указанным временем (2s) пока вход I0.0 равен 1. Если состояние сигнала на входе I0.0 меняется с 1 на 0 до истечения заданного времени, таймер останавливается. Если состояние сигнала на входе I0.1 меняется с 0 на 1 во время работы таймера, то таймер сбрасывается. Состояние сигнала Q4.0 равно 1 пока таймер работает и равно 0 по окончании отсчета времени или при сбросе таймера .

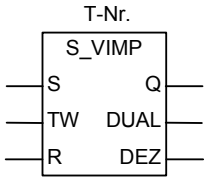
13.4 S_PEXT : : Задание параметров и запуск таймера «Удлиненный импульс»

Обозначение

Английский



Немецкий



Параметры Английский	Параметры Немецкий	Тип данных	Область памяти	Описание
T no.	T Nr.	TIMER	T	Номер таймера. Диапазон номеров зависит от CPU.
S	S	BOOL	I, Q, M, D, L	Вход запуска
TV	TW	S5TIME	I, Q, M, D, L	Установка времени
R	R	BOOL	I, Q, M, D, L	Вход сброса
BI	DUAL	WORD	I, Q, M, D, L	Остаток времени (значение в двоичном коде)
BCD	DEZ	WORD	I, Q, M, D, L	Остаток времени (значение в формате BCD)
Q	Q	BOOL	I, Q, M, D, L	Состояние таймера

Описание

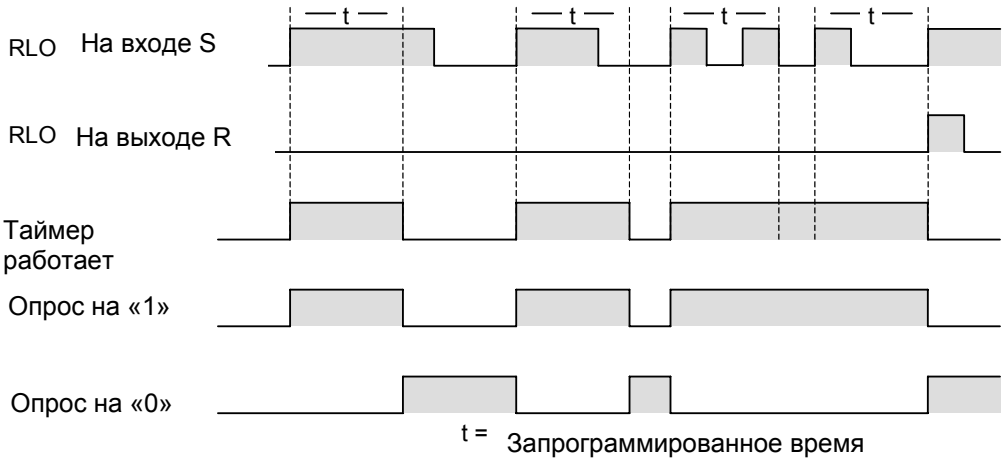
S_PEXT:(S5 таймер «Удлиненный Импульс») запускает заданный таймер, по нарастающему фронту на входе запуска (S). Для запуска таймера всегда необходимо изменение сигнала. Таймер продолжает работать в течение времени, заданного на входе TV, даже если состояние сигнала на входе S меняется на 0 до истечения заданного времени. Пока таймер работает выход Q выдает сигнал 1. Таймер перезапускается с заданным временем, если состояние сигнала на входе S меняется с 0 на 1 во время работы таймера.

Если во время работы таймера происходит изменение с 0 на 1 сигнала на входе сброса (R), то таймер сбрасывается. Это изменение сбрасывает в ноль время и базу времени.

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на BI представлено в двоичном формате, а на BCD - в двоично-десятичном формате. Текущее время равно разнице между начальным значением, заданным на входе TV и временем , прошедшим с момента запуска таймера.

Временные диаграммы

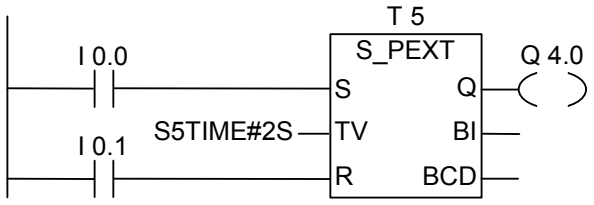
Таймер удлиненный импульс:



Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	X	X	X	1

Пример

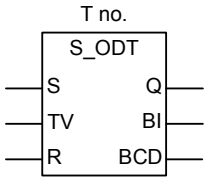


Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт RLO), таймер T5 запускается. Таймер продолжает работать указанное время (2s) даже после сброса сигнала на входе S в 0. Если на входе I0.0 вновь появляется нарастающий фронт RLO до истечения заданного времени, таймер перезапускается. Если состояние сигнала на входе I0.1 меняется 0 на 1 во время работы таймера, то таймер сбрасывается. Состояние сигнала на выходе Q4.0 равно 1,пока таймер работает.

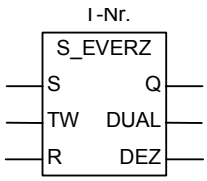
13.5 S_ODT : Задание параметров и запуск таймера «Задержка включения»

Обозначение

Английский



Немецкий



Параметры Английский	Параметры Немецкий	Тип данных	Область памяти	Описание
T no.	T Nr.	TIMER	T	Номер таймера. Диапазон номеров зависит от CPU.
S	S	BOOL	I, Q, M, D, L	Вход запуска
TV	TW	S5TIME	I, Q, M, D, L	Установка времени
R	R	BOOL	I, Q, M, D, L	Вход сброса
BI	DUAL	WORD	I, Q, M, D, L	Остаток времени (значение в двоичном коде)
BCD	DEZ	WORD	I, Q, M, D, L	Остаток времени (значение в формате BCD)
Q	Q	BOOL	I, Q, M, D, L	Состояние таймера

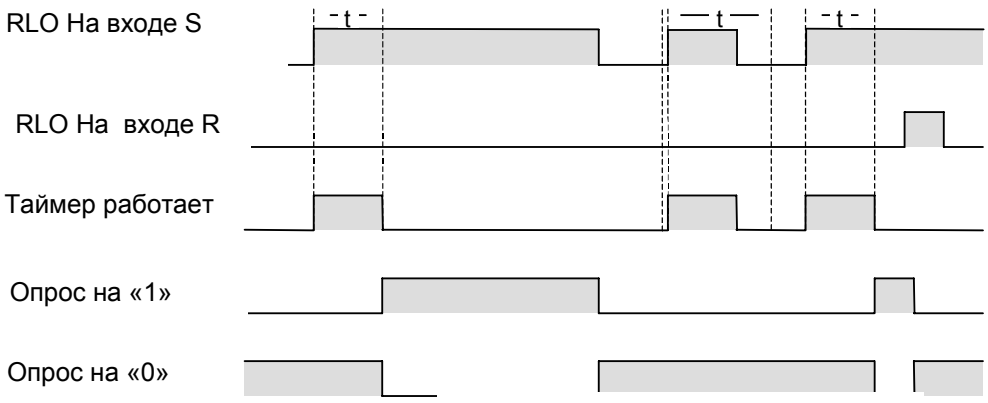
Описание

S_ODT: (**таймер «Задержка включения»)** запускает заданный таймер если имеется нарастающий фронт (изменение состояния сигнала с 0 на 1) на входе запуска (S). Для запуска таймера всегда необходим фронт сигнала. Таймер продолжает работать в течение времени, заданного на входе TV, пока состояние сигнала на входе S равно 1. Выход Q выдает 1, когда время истекло без ошибок при состоянии сигнала на входе S равном 1. Если состояние сигнала на входе S меняется с 1 на 0 во время работы таймера, таймер останавливается. В этом случае выход Q остается в 0. Если во время работы таймера происходит изменение с 0 на 1 сигнала на входе сброса (R), то таймер сбрасывается. Это изменение сбрасывает в ноль время и базу времени. На выходе Q выдается 0. При этом таймер сбрасывается независимо от сигнала на входе S .

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на BI представлено в двоичном формате, а на BCD - в двоично-десятичном формате. Это время равно разнице между начальным значением, заданным на входе TV и временем , прошедшим с момента запуска таймера.

Временные диаграммы

Таймер с задержкой включения:

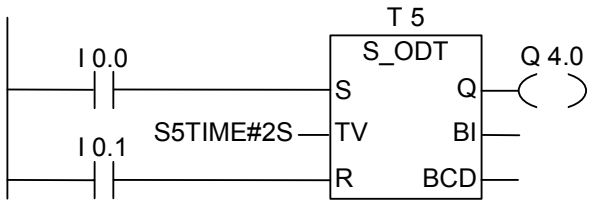


T - Заданное время

Биты слова состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	X	X	X	1

Пример

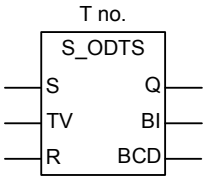


Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт RLO), таймер T5 запускается. Если заданное время (2s) истекло и состояние сигнала на входе I0.0 равно 1, то состояние сигнала на выходе Q4.0 становится равным 1. Если состояние сигнала на входе I0.0 меняется с 1 на 0, то таймер останавливается и выход Q4.0 равен 0 (если состояние сигнала на входе I0.1 меняется с 0 на 1, время сбрасывается независимо от того – работал таймер или нет).

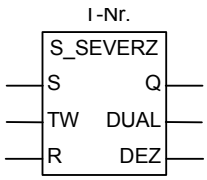
13.6 S_ODTS : Задание параметров и запуск таймера «Задержка включения с памятью»

Обозначение

Английский



Немецкий



Параметры Английский	Параметры Немецкий	Тип данных	Область памяти	Описание
T no.	T Nr.	TIMER	T	Номер таймера. Диапазон номеров зависит от CPU.
S	S	BOOL	I, Q, M, D, L	Вход запуска
TV	TW	S5TIME	I, Q, M, D, L	Установка времени
R	R	BOOL	I, Q, M, D, L	Вход сброса
BI	DUAL	WORD	I, Q, M, D, L	Остаток времени (значение в двоичном коде)
BCD	DEZ	WORD	I, Q, M, D, L	Остаток времени (значение в формате BCD)
Q	Q	BOOL	I, Q, M, D, L	Состояние таймера

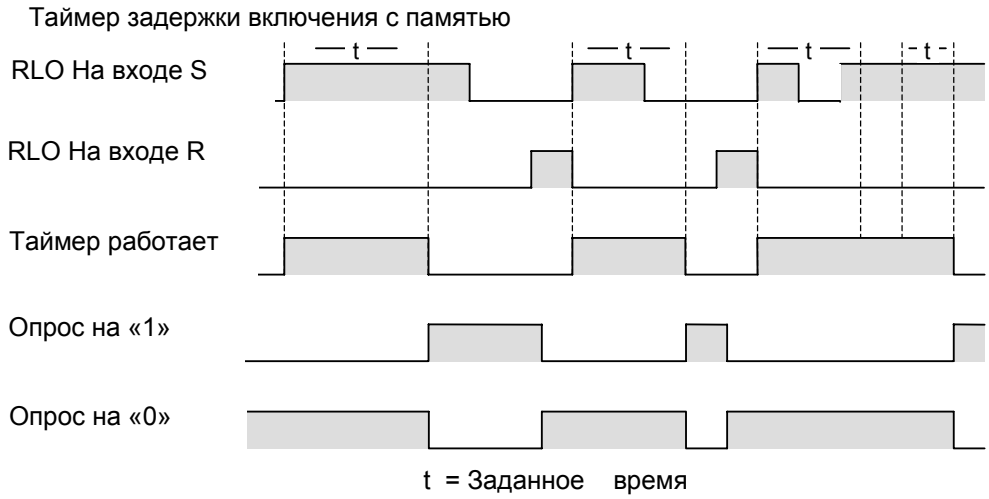
Описание

S_ODTS: (Таймер «Задержка включения с памятью») запускает заданный таймер, если имеется нарастающий фронт на входе запуска (S). Для запуска таймера всегда необходимо изменение сигнала. Таймер продолжает работать в течение времени, заданного на входе TV, даже если состояние сигнала на входе S меняется на 0 до истечения заданного времени. Состояние сигнала на выходе Q выдает значение 1, по истечении заданного времени, независимо от состояния сигнала на входе S. 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 сигнала на входе сброса (R) таймера сбрасывает таймер независимо от состояния RLO на входе S.

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на BI представлено в двоичном формате, а на BCD - в двоично-десятичном формате. Это время равно разнице между начальным значением, заданным на входе TV и временем , прошедшим с момента запуска таймера.

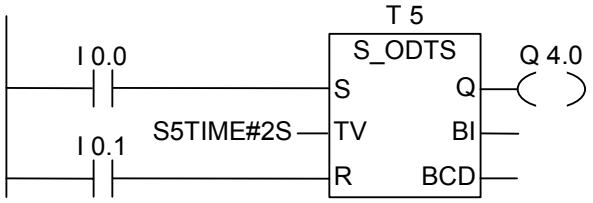
Временные диаграммы



Биты слова состояния

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

Пример



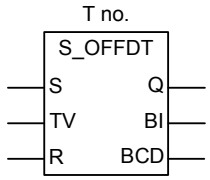
Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт RLO), таймерT5 запускается .Таймер продолжает работат несмотря на изменение сигнала на входе I0.0 с 1 на 0. Если состояние сигнала на входе I0.0 меняется с 0 на 1 до истечения заданного времени, таймер перезапускается.

Состояние сигнала на выходе Q4.0 устанавливается в 1 по истечении заданного времени . (Если состояние сигнала на входе I0.1 меняется с 0 на 1, время сбрасывается независимо от RLO на входе S).

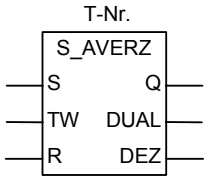
13.7 S_OFFDT : Задание параметров и запуск таймера «Задержка выключения»

Обозначение

Английский



Немецкий



Параметры Английский	Параметры Немецкий	Тип данных	Область памяти	Описание
T no.	T Nr.	TIMER	T	Номер таймера . Диапазон номеров зависит от CPU.
S	S	BOOL	I, Q, M, D, L	Вход запуска
TV	TW	S5TIME	I, Q, M, D, L	Установка времени
R	R	BOOL	I, Q, M, D, L	Вход сброса
BI	DUAL	WORD	I, Q, M, D, L	Остаток времени (значение в двоичном коде)
BCD	DEZ	WORD	I, Q, M, D, L	Остаток времени (значение в формате BCD)
Q	Q	BOOL	I, Q, M, D, L	Состояние таймера

Описание

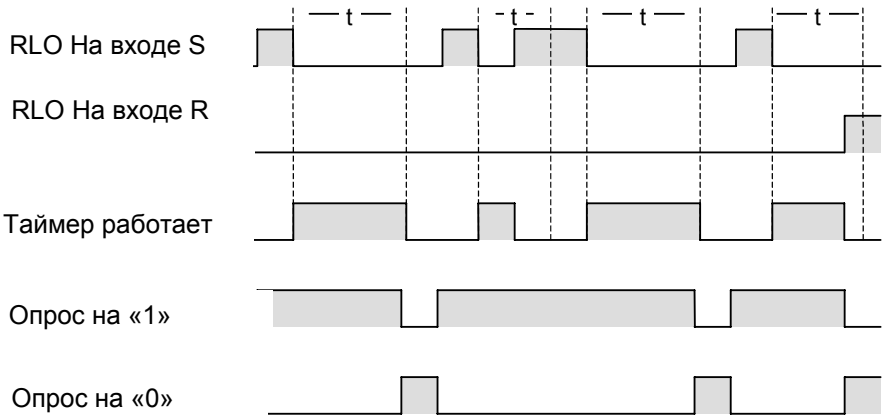
S_OFFDT : (Таймер «Задержка выключения») запускает заданный таймер, если имеется падающий фронт на входе запуска (S). Для запуска таймера всегда необходимо изменение сигнала. Выход Q равен 1, когда состояние сигнала на входе S равно 1 или пока таймер работает. Таймер сбрасывается, когда состояние сигнала на входе S меняется с 0 на 1 во время работы таймера. Таймер не перезапускается, пока состояние сигнала на входе S снова не изменится с 1 на 0.

Изменение с 0 на 1 сигнала на входе сброса (R) таймера во время его работы сбрасывает таймер.

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на BI представлено в двоичном формате, а на BCD - в двоично-десятичном формате. Это время равно разнице между начальным значением, заданным на входе TV и временем , прошедшим с момента запуска таймера.

Временные диаграммы

Таймер с задержкой выключения:

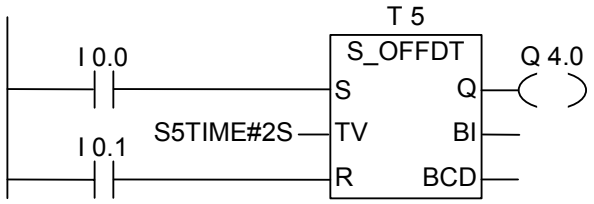


t =Заданное время

Биты слова состояния

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

Пример



Если состояние сигнала на входе I0.0 меняется с 1 на 0, таймер запускается. Выход Q4.0 равен 1 когда I0.0 равен 1 или таймер работает. Если состояние сигнала на I0.1 меняется с 0 на 1 когда таймер работает, то таймер сбрасывается.

13.8 ---(SP) Запуск таймера «Импульс»

Обозначение

Английский	**Немецкий**
<T no.>	<T no.>
---(SP)	**---(SI)**
<Значение времени>	< Значение времени >

Параметр	**Тип данных**	**Область памяти**	**Описание**
<T no.>	TIMER	T	Номер таймера .Диапазон номеров зависит от CPU.
< Значение времени >	S5TIME	I, Q, M, L, D	Заданное значение времени

Описание

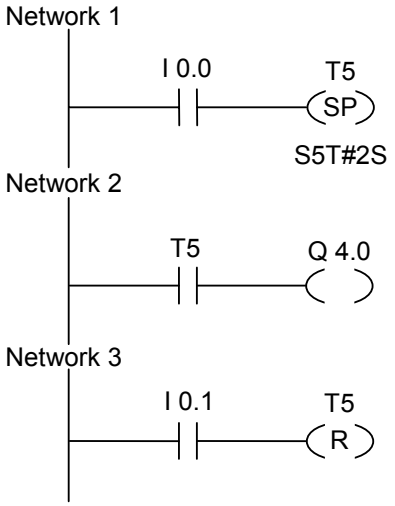
---(SP) (**Запуск таймера «Импульс»)** запускает заданный таймер со **< Значением времени >** по нарастающему фронту на входе запуска. Пока RLO на входе запуска положительный (“1”), таймер продолжает отсчитывать заданное время. Пока таймер работает, опрос таймера на состояние 1 дает результат опроса 1. Если на входе запуска сигнал меняется с 1 на 0 до истечения заданного времени, таймер останавливается . Тогда опрос таймера на состояние 1 дает результат опроса 0.

Дополнительную информацию Вы найдете в руководстве "Location of a Timer in Memory and Components of a Timer"(Расположение таймеров в памяти и компонентов таймера) и описании работы таймера S_PULSE.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
Записывает	-	-	-	-	-	0	-	-	0

Пример



Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт RLO), таймер T5 запускается. Таймер продолжает работать в течение указанного времени (2s) пока вход I0.0 равен 1. Если состояние сигнала на входе I0.0 меняется с 1 на 0 до истечения заданного времени, таймер останавливается.

Состояние сигнала Q4.0 равно 1 пока таймер работает. Если состояние сигнала на входе I0.1 меняется с 0 на 1 , то таймер T5 останавливается и сбрасывает оставшееся значение времени в 0.

13.9 ---(SE) : Запуск таймера «Удлиненный импульс»

Обозначение

Английский	**Немецкий**
<T no.>	<T no>
---(SE)	**---(SV)**
< Значение времени >	< Значение времени >

Параметр	**Тип данных**	**Область памяти**	**Описание**
<T no.>	TIMER	T	Номер таймера .Диапазон номеров зависит от CPU.
< Значение времени >	S5TIME	I, Q, M, L, D	Заданное значение времени

Описание

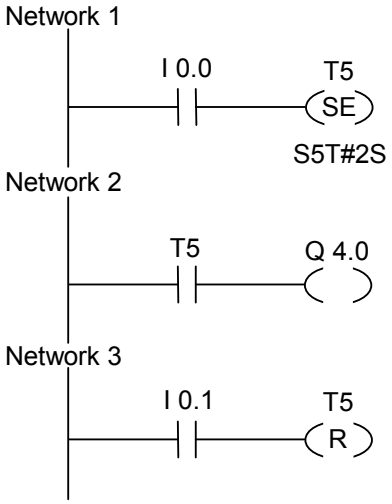
---(SE) (**Запуск таймера «Удлиненный импульс»)** запускает заданный таймер на **< Значение времени >** , если имеется нарастающий фронт на входе запуска. Таймер продолжает работать в течение времени, заданного на входе TV, даже если состояние сигнала на входе запуска меняется на 0 до истечения заданного времени. Пока таймер работает, опрос таймера на состояние 1 выдает результат 1. Таймер перезапускается с заданным временем, если состояние сигнала на входе S меняется с 0 на 1 во время работы таймера

Дополнительную информацию Вы найдете в руководстве "Location of a Timer in Memory and Components of a Timer"(Расположение таймеров в памяти и компонентов таймера) и описании работы таймера S_PEXT.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
Записывает	-	-	-	-	-	0	-	-	0

Пример



Если состояние сигнала на входе I0.0 меняется 0 на 1 (нарастающий фронт RLO), таймер T5 запускается. Таймер продолжает работать указанное время (2s) даже при появлении падающего фронта сигнала на входе запуска. Если на входе I0.0 вновь появляется нарастающий фронт RLO до истечения заданного времени, таймер перезапускается. Состояние сигнала на выходе Q4.0 равно 1 пока таймер работает. Смена состояния сигнала на входе I0.1 сбрасывает таймер T5 и сбрасывает оставшееся значение времени в 0.

13.10 ---(SD) : Запуск таймера «Задержка включения»

Обозначение

Английский	**Немецкий**
<T no.>	<T no.>
---(SD)	**---(SE)**
< Значение времени >	< Значение времени >

Параметр	**Тип данных**	**Область памяти**	**Описание**
<T no.>	TIMER	T	Номер таймера .Диапазон номеров зависит от CPU.
< Значение времени >	S5TIME	I, Q, M, L, D	Заданное значение времени

Описание

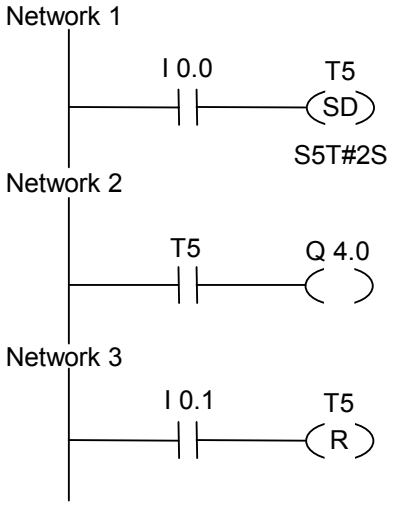
---(SD) (Запуск таймера «Задержка включения») запускает заданный таймер на **< Значение времени >** , если имеется нарастающий фронт на входе запуска. Таймер получает состояние выхода 1 по окончании отсчета заданного времени и при RLO = 1на входе . Если состояние сигнала на входе запуска меняется с 1 на 0 во время работы таймера, таймер сбрасывается. В этом случае опрос таймера на состояние 1 выдает результат опроса 0.

Дополнительную информацию Вы найдете в руководстве "Location of a Timer in Memory and Components of a Timer"(Расположение таймеров в памяти и компонентов таймера) и описании работы таймера S_ODT.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
Записывает	-	-	-	-	-	0	X	X	0

Пример



Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт RLO), таймерT5 запускается. Если заданное время (2s) истекло и состояние сигнала на входе I0.0 равно 1, то состояние сигнала на выходе Q4.0 становится равным 1.

Если состояние сигнала на входе I0.0 меняется с 1 на 0, то таймер останавливается и выход Q4.0 равен 0. Если состояние сигнала на входе I0.1 меняется с 0 на 1, при работе таймера, то таймер T5 останавливается и оставшееся значение времени сбрасывается в 0.

13.11 ---(SS) : Запуск таймера «Задержка включения с памятью»

Обозначение

Английский	**Немецкий**
<T no.>	<T no.>
---(SS)	**---(SS)**
<Значение времени>	< Значение времени >

Параметр	**Тип данных**	**Область памяти**	**Описание**
<T no.>	TIMER	T	Номер таймера .Диапазон номеров зависит от CPU.
< Значение времени >	S5TIME	I, Q, M, L, D	Заданное значение времени

Описание

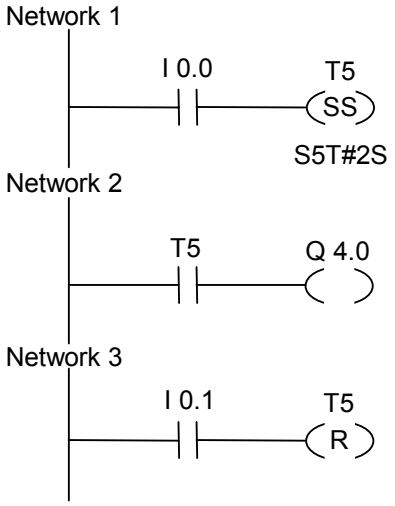
---(SS) : (Запуск таймера «Задержка включения с памятью») запускает заданный таймер, если имеется нарастающий фронт на входе запуска . Статус таймера принимает значение 1 по истечении заданного времени. Перезапуск таймера возможен только при его сбросе. Только сброс таймера позволяет перевести его в состояние 0. Таймер перезапускается на заданное время, если состояние сигнала на входе запуска меняется с 0 на 1 во время работы таймера.

Дополнительную информацию Вы найдете в руководстве "Location of a Timer in Memory and Components of a Timer"(Расположение таймеров в памяти и компонентов таймера) и описании работы таймера S_ODT.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
Записывает	-	-	-	-	-	0	X	X	0

Пример



Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт RLO), таймерT5 запускается . Если состояние сигнала на входе I0.0 меняется с 0 на 1 до истечения заданного времени, таймер перезапускается. Состояние сигнала на выходе Q4.0 устанавливается в 1 если заданное время истекло. Состояние сигнала на входе I0.1 = 1 сбрасывает таймер T5 , стирая оставшееся время.

13.12 ---(SF) : Запуск таймера «Задержка выключения»

Обозначение

Английский	**Немецкий**
<T no.>	<T no.>
---(SF)	**---(SA)**
<Значение времени>	< Значение времени >

Параметр	**Тип данных**	**Область памяти**	**Описание**
<T no.>	TIMER	T	Номер таймера .Диапазон номеров зависит от CPU.
< Значение времени >	S5TIME	I, Q, M, L, D	Заданное значение времени

Описание

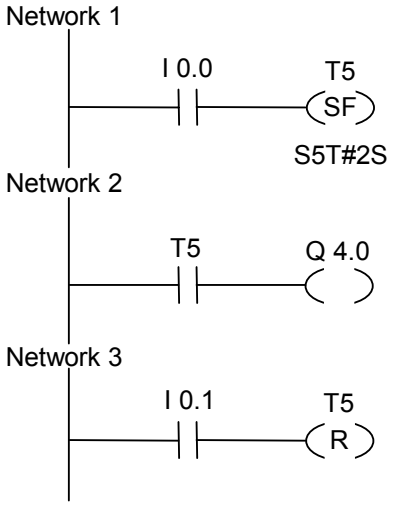
---(SF) : (**«Задержка выключения»)** запускает заданный таймер, по падающему фронту на входе запуска . Таймер имеет состояние 1 , когда состояние сигнала на входе запуска равно 1 или пока таймер производит отсчет заданного **< Значения времени >**. Таймер сбрасывается, когда состояние сигнала на входе запуска меняется с 0 на 1 во время работы таймера. Таймер всегда перезапускается при изменении состояния сигнала на входе запуска с 1 на 0.

Дополнительную информацию Вы найдете в руководстве "Location of a Timer in Memory and Components of a Timer"(Расположение таймеров в памяти и компонентов таймера) и описании работы таймера SF.

Биты слова состояния

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
Записывает	-	-	-	-	-	0	-	-	0

Пример



Если состояние сигнала на входе I0.0 меняется с 1 на 0, таймер запускается. Выход Q4.0 равен 1 когда I0.0 равен 1 или таймер работает. Если состояние сигнала на входе I0.1 меняется с 0 на 1 когда таймер работает, то таймер сбрасывается и остаток времени обнуляется.

14 Поразрядные логические инструкции со словами

14.1 Обзор логических инструкций над словами

Описание

Поразрядные логические инструкции над словами комбинируют пары слов (16 бит) или двойных слов (32 бита) побитно в соответствии с правилами булевой логики.

Если значение результата на выходе OUT не равно 0, бит CC 1 в слове состояния устанавливается в 1.

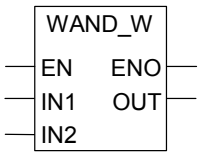
Если значение результата на выходе OUT равно 0, бит CC 1 в слове состояния сбрасывается в 0.

Для выполнения логических операций над словами предоставляются следующие инструкции:

- WAND_W : Поразрядное И над словами
- WOR_W : Поразрядное ИЛИ над словами
- WXOR_W : Поразрядное исключающее ИЛИ над словами
- WAND_DW : Поразрядное И над двойными словами
- WOR_DW : Поразрядное ИЛИ над двойными словами
- WXOR_DW : Поразрядное исключающее ИЛИ над двойными словами

14.2 WAND_W : Поразрядное И над словами

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	WORD	I, Q, M, L, D	Первое значение для логической операции
IN2	WORD	I, Q, M, L, D	Второе значение для логической операции
OUT	WORD	I, Q, M, L, D	Результат выполнения инструкции

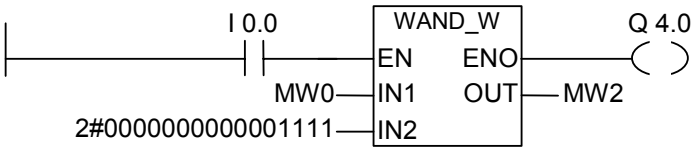
Описание

WAND_W: (**Поразрядное И над словами)** активируется при подаче 1 на вход деблокировки (EN) .Эта команда производит функцию побитового И над двумя словами на входах IN1 и IN2.Эти значения интерпретируются как простые последовательности битов. Результат может быть считан на выходе OUT. ENO имеет одинаковое состояние сигнала с EN.

Биты байта состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	1	X	0	0	-	X	1	1	1

Пример



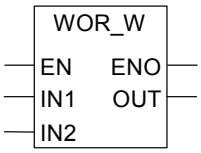
Инструкция активируется состоянием сигнала 1 на входе I0.0. Только биты с 0 по 3 будут сохранены, все остальные биты MW0 обнуляются.

MW0	=	0101010101010101
IN2	=	0000000000001111
MW0 **AND** IN2 = MW2	=	0000000000000101

Если инструкция выполнена , то выход Q4.0 =1

14.3 WOR_W : Поразрядное ИЛИ над словами

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	WORD	I, Q, M, L, D	Первое значение для логической операции
IN2	WORD	I, Q, M, L, D	Второе значение для логической операции
OUT	WORD	I, Q, M, L, D	Результат выполнения инструкции

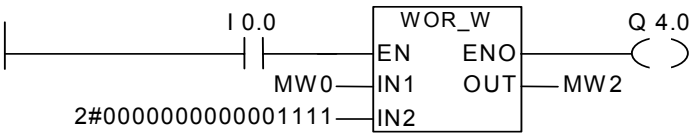
Описание

WOR_W :(**Поразрядное И над словами)** активируется при подаче 1 на вход деблокировки (EN) .Эта команда производит функцию побитового ИЛИ над двумя словами на входах IN1 и IN2.Эти значения интерпретируются как простые последовательности битов. Результат может быть считан на выходе OUT. ENO имеет одинаковое состояние сигнала с EN.

Биты байта состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	1	X	0	0	-	X	1	1	1

Пример



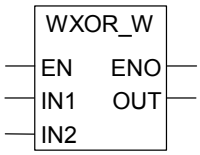
Инструкция активируется состоянием сигнала 1 на входе I0.0. При этом биты с 0 по 3 будут установлены в 1, все остальные биты MW0 останутся неизменными и будут переданы в MW2.

MW0	=	0101010101010101
IN2	=	0000000000001111
MW0 **OR** IN2 =MW2	=	0101010101011111.

Выход Q4.0 =1 если инструкция выполнилась.

14.4 WXOR_W : Поразрядное исключающее ИЛИ над словами

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	WORD	I, Q, M, L, D	Первое значение для логической операции
IN2	WORD	I, Q, M, L, D	Второе значение для логической операции
OUT	WORD	I, Q, M, L, D	Результат выполнения инструкции

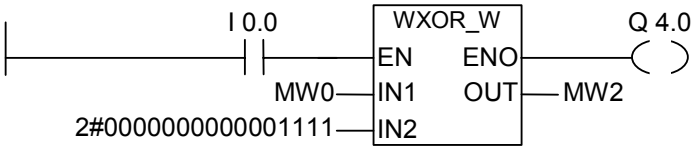
Описание

WXOR_W: (Поразрядное исключающее ИЛИ над словами) активируется при подаче 1 на вход деблокировки (EN) .Эта команда производит функцию побитового исключающего ИЛИ над двумя словами на входах IN1 и IN2.Эти значения интерпретируются как простые последовательности битов. Результат может быть считан на выходе OUT. ENO имеет одинаковое состояние сигнала с EN.

Биты байта состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	1	X	0	0	-	X	1	1	1

Пример



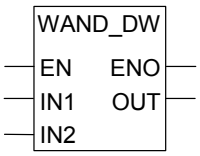
Инструкция активируется при I0.0 = 1.

MW0	=	0101010101010101
IN2	=	0000000000001111
MW0 **XOR** IN2=MW2	=	0101010101011010

Q4.0 =1 если инструкция выполняется.

14.5 WAND_DW : Поразрядное И над двойным словом

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	DWORD	I, Q, M, L, D	Первое значение для логической операции
IN2	DWORD	I, Q, M, L, D	Второе значение для логической операции
OUT	DWORD	I, Q, M, L, D	Результат выполнения инструкции

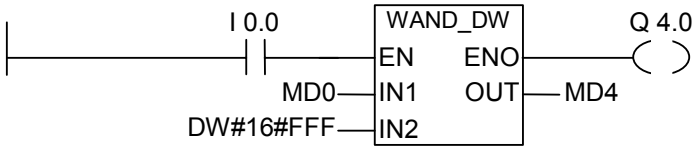
Описание

WAND_DW: (**Поразрядное И над двойными словами)** активируется при подаче 1 на вход деблокировки (EN) .Эта команда производит функцию побитового И над двумя двойными словами на входах IN1 и IN2.Эти значения интерпретируются как простые последовательности битов. Результат может быть считан на выходе OUT. ENO имеет одинаковое состояние сигнала с EN.

Биты байта состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	1	X	0	0	-	X	1	1	1

Пример

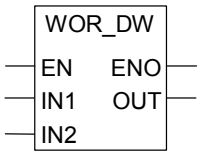


Инструкция выполняется при I0.0 = 1. Только биты с 0 по 11 сохраняются, все остальные биты MD4 обнуляются. Q4.0 = 1 если инструкция выполняется.

MD0	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
MD0 **AND** IN2 =MD4	=	0000000000000000	0000010101010101

14.6 WOR_DW : Поразрядное ИЛИ над двойными словами

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	DWORD	I, Q, M, L, D	Первое значение для логической операции
IN2	DWORD	I, Q, M, L, D	Второе значение для логической операции
OUT	DWORD	I, Q, M, L, D	Результат выполнения инструкции

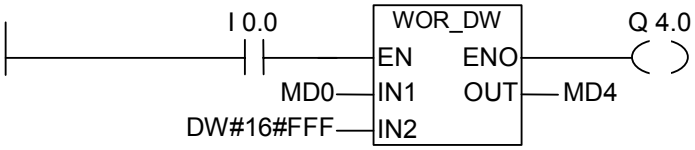
Описание

WOR_DW: (**Поразрядное ИЛИ над двойными словами**)
активируется при подаче 1 на вход деблокировки (EN) .Эта команда производит функцию побитового ИЛИ над двумя двойными словами, поданными на входы IN1 и IN2. Эти значения интерпретируются как простые последовательности битов. Результат может быть считан на выходе OUT. ENO имеет одинаковое состояние сигнала с EN.

Биты байта состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	1	X	0	0	-	X	1	1	1

Пример

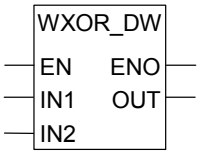


Инструкция выполняется при I0.0 = 1. Биты с 0 по 11 устанавливаются в 1, остальные передаются в MD4 без изменений. Выход Q4.0 = 1 при выполнении функции.

MD0	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
MD0 **OR** IN2=MD4	=	0101010101010101	0101111111111111

14.7 WXOR_DW : Поразрядное Исключающее ИЛИ над двойными словами

Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D	Деблокировка входа
ENO	BOOL	I, Q, M, L, D	Деблокировка выхода
IN1	DWORD	I, Q, M, L, D	Первое значение для логической операции
IN2	DWORD	I, Q, M, L, D	Второе значение для логической операции
OUT	DWORD	I, Q, M, L, D	Результат выполнения инструкции

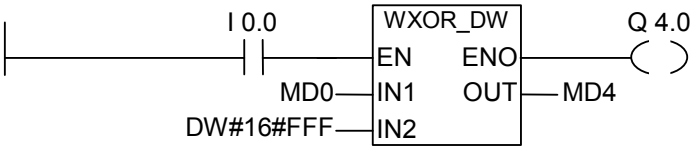
Описание

WXOR_DW: (Поразрядное исключающее ИЛИ над двойными словами) активируется при подаче 1 на вход деблокировки (EN) .Эта команда выполняет функцию побитового исключающего ИЛИ над двумя двойными словами, поданными на входы IN1 и IN2. Эти значения интерпретируются как простые последовательности битов. Результат может быть считан на выходе OUT. ENO имеет одинаковое состояние сигнала с EN.

Биты байта состояния

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	1	X	0	0	-	X	1	1	1

Пример



Инструкция выполняется при I0.0 = 1, при этом Q4.0 = 1.

MD0	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
MD0 **XOR** IN2 = MD4	=	0101010101010101	0101101010101010

A Обзор всех KOP инструкций

A.1 KOP инструкции, в алфавитном порядке английской мнемоники (International)

Английская мнемоника	Немецкая мнемоника	Каталог программных элементов	Описание
---| |---	---| |---	Bit logic Instruction	Нормально открытый контакт (адрес)
---|/|---	---|/|---	Bit logic Instruction	Нормально замкнутый контакт (адрес)
---( )	---( )	Bit logic Instruction	Выходная катушка
---(#)---	---(#)---	Bit logic Instruction	Промежуточный результат (Коннектор)
==0 ---| |---	==0 ---| |---	Status bits	Результат опроса на “равно 0”
>0 ---| |---	>0 ---| |---	Status bits	Результат опроса на “больше 0”
>=0 ---| |---	>=0 ---| |---	Status bits	Результат опроса на “больше или равно 0”
<=0 ---| |---	<=0 ---| |---	Status bits	Результат опроса на “меньше или равно 0”
<0 ---| |---	<0 ---| |---	Status bits	Результат опроса на “меньше 0”
<>0 ---| |---	<>0 ---| |---	Status bits	Результат опроса на “не равно 0”
ABS	ABS	Floating point Instruction	Вычисление абсолютной величины числа с плавающей точкой
ACOS	ACOS	Floating point Instruction	Вычисление арккосинуса угла для числа с плавающей точкой
ADD_DI	ADD_DI	Integer Math Instruction	Сложение двойных целых чисел
ADD_I	ADD_I	Integer Math Instruction	Сложение целых чисел
ADD_R	ADD_R	Floating point Instruction	Сложение чисел с плавающей точкой
ASIN	ASIN	Floating point Instruction	Вычисление арксинуса угла для числа с плавающей точкой
ATAN	ATAN	Floating point Instruction	Вычисление арктангенса угла для числа с плавающей точкой
BCD_DI	BCD_DI	Convert	Преобразовать двоично-десятичное число в двойное целое
BCD_I	BCD_I	Convert	Преобразовать двоично-десятичное число в целое
BR ---| |---	BIE ---| |---	Status bits	Бит ошибки "Регистр BR"
----(CALL)	----(CALL)	Program control	Вызов FC или SFC (без параметров)
CALL_FB	CALL_FB	Program control	Вызвать FB из списка блоков
CALL_FC	CALL_FC	Program control	Вызвать FC из списка блоков

Английская мнемоника	**Немецкая мнемоника**	**Каталог программных элементов**	**Описание**
CALL_SFB	CALL_SFB	Program control	Вызвать SFB из списка блоков
CALL_SFC	CALL_SFC	Program control	Вызвать SFC из списка блоков
----(CD)	----(ZR)	Counters	Катушка счетчика на уменьшение
CEIL	CEIL	Convert	Округлить до ближайшего большего целого числа
CMP >=D	CMP >=D	Compare	Сравнение двойных целых чисел (==, <>, >, <, >=, <=)
CMP >=I	CMP >=I	Compare	Сравнение целых чисел (==, <>, >, <, >=, <=)
CMP >=R	CMP >=R	Compare	Сравнение чисел с плавающей точкой (==, <>, >, <, >=, <=)
COS	COS	Floating point Instruction	Вычисление косинуса
----(CU)	---(ZV)	Counters	Катушка счетчика на увеличение
DI_BCD	DI_BCD	Convert	Преобразовать двойное целое число в двоично-десятичное
DI_R	DI_R	Convert	Преобразовать двойное целое число в число с плавающей точкой
DIV_DI	DIV_DI	Integer Math Instruction	Деление двойных целых чисел
DIV_I	DIV_I	Integer Math Instruction	Деление целых чисел
DIV_R	DIV_R	Floating point Instruction	Деление чисел с плавающей точкой
EXP	EXP	Floating point Instruction	Вычисление экспоненты
FLOOR	FLOOR	Convert	Округлить до ближайшего меньшего целого числа
I_BCD	I_BCD	Convert	Преобразовать целое число в двоично-десятичное
I_DI	I_DI	Convert	Преобразовать целое число в двойное целое
INV_I	INV_I	Convert	Инверсия целого числа
INV_DI	INV_DI	Convert	Инверсия двойного целого числа
---(JMP)	---(JMP)	Jumps	Безусловный переход
---(JMP)	---(JMP)	Jumps	Условный переход
---(JMPN)	---(JMPN)	Jumps	Переход по нулевому результату
LABEL	LABEL	Jumps	Метка
LN	LN	Floating point Instruction	Вычисление натурального логарифма
---(MCR>)	---(MCR>)	Program control	Выключить главное управляющее реле
---(MCR<)	---(MCR<)	Program control	Включить главное управляющее реле
---(MCRA)	---(MCRA)	Program control	Активировать главное управляющее реле
---(MCRD)	---(MCRD)	Program control	Деактивировать главное управляющее реле
MOD_DI	MOD_DI	Integer Math Instruction	Выделение остатка от деления двойных целых чисел
MOVE	MOVE	Move	Передача значения
MUL_DI	MUL_DI	Integer Math Instruction	Умножение двойных целых чисел

Английская мнемоника	**Немецкая мнемоника**	**Каталог программных элементов**	**Описание**
MUL_I	MUL_I	Integer Math Instruction	Умножение целых чисел
MUL_R	MUL_R	Floating point Instruction	Умножение чисел с плавающей точкой
---(N)---	---(N)---	Bit logic Instruction	Выделение отрицательного фронта RLO
NEG	NEG	Bit logic Instruction	Выделение отрицательного фронта сигнала
NEG_DI	NEG_DI	Convert	Дополнительный код двойного целого числа
NEG_I	NEG_I	Convert	Дополнительный код целого числа
NEG_R	NEG_R	Convert	Изменение знака числа с плавающей точкой
---| NOT |---	---| NOT |---	Bit logic Instruction	Инвертировать результат логической операции (поток энергии)
---(OPN)	---(OPN)	DB call	Открыть блок данных : DB или DI
OS ---| |---	OS ---| |---	Status bits	Бит ошибки переполнение с памятью
OV ---| |---	OV ---| |---	Status bits	Бит ошибки переполнение
---(P)---	---(P)---	Bit logic Instruction	Выделение положительного фронта RLO
POS	POS	Bit logic Instruction	Выделение положительного фронта сигнала
---(R)	---(R)	Bit logic Instruction	Сбросить выход в 0
---(RET)	---(RET)	Program control	Возврат
ROL_DW	ROL_DW	Shift/Rotate	Циклический сдвиг двойного целого числа влево
ROR_DW	ROR_DW	Shift/Rotate	Циклический сдвиг двойного целого числа вправо
ROUND	ROUND	Convert	Округлить до двойного целого числа
RS	RS	Bit logic Instruction	RS -триггер
---(S)	---(S)	Bit logic Instruction	Установить выход в 1
---(SAVE)	---(SAVE)	Bit logic Instruction	Сохранение бита RLO в BR бите
---(SC)	---(SZ)	Counters	Установить значение счетчика
S_CD	Z_RUECK	Counters	Счетчик обратного счета
S_CU	Z_VORW	Counters	Счетчик прямого счета
S_CUD	ZAEHLER	Counters	Счетчик прямого и обратного счета
---(SD)	---(SE)	Timers	Катушка таймер задержки включения
---(SE)	---(SV)	Timers	Катушка удлиненного импульса
---(SF)	---(SA)	Timers	Катушка задержки выключения
SHL_DW	SHL_DW	Shift/Rotate	Сдвиг двойного слова влево
SHL_W	SHL_W	Shift/Rotate	Сдвиг слова влево
SHR_DI	SHR_DI	Shift/Rotate	Сдвиг двойного целого слова вправо
SHR_DW	SHR_DW	Shift/Rotate	Сдвиг двойного слова вправо
SHR_I	SHR_I	Shift/Rotate	Сдвиг целого слова вправо
SHR_W	SHR_W	Shift/Rotate	Сдвиг слова вправо
SIN	SIN	Floating point Instruction	Вычисление синуса угла числа с плавающей точкой
S_ODT	S_EVERZ	Timers	S5 таймер задержки включения
S_ODTS	S_SEVERZ	Timers	S5 таймер задержки включения с памятью

Английская мнемоника	**Немецкая мнемоника**	**Каталог программных элементов**	**Описание**
S_OFFDT	S_AVERZ	Timers	S5 таймер задержки выключения
---(SP)	---(SI)	Timers	Катушка таймера “Импульс”
S_PEXT	S_VIMP	Timers	S5 таймер “Импульс с памятью”
S_PULSE	S_IMPULS	Timers	S5 таймер “Импульс”
SQR	SQR	Floating point Instruction	Вычисление квадрата числа с плавающей точкой
SQRT	SQRT	Floating point Instruction	Вычисление квадратного корня числа с плавающей точкой
SR	SR	Bit logic Instruction	SR- триггер
---(SS)	---(SS)	Timers	Катушка таймера задержки включени я с памятью
SUB_DI	SUB_DI	Integer Math Instruction	Вычитание двойных целых чисел
SUB_I	SUB_I	Integer Math Instruction	Вычитание целых чисел
SUB_R	SUB_R	Floating point Instruction	Вычитание чисел с плавающей точкой
TAN	TAN	Floating point Instruction	Вычисление тангенса числа с плавающей точкой
TRUNC	TRUNC	Convert	Выделение целой части числа
UO ---| |---	UO ---| |---	Status bits	Бит ошибки недопустимой операции
WAND_DW	WAND_DW	Word logic Instruction	Поразрядное И над двойными словами
WAND_W	WAND_W	Word logic Instruction	Поразрядное И над словами
WOR_DW	WOR_DW	Word logic Instruction	Поразрядное ИЛИ над двойными словами
WOR_W	WOR_W	Word logic Instruction	Поразрядное ИЛИ над словами
WXOR_DW	WXOR_DW	Word logic Instruction	Поразрядное исключающее ИЛИ над двойными словами
WXOR_W	WXOR_W	Word logic Instruction	Поразрядное исключающее ИЛИ над словами

A.2 КОР инструкции, в алфавитном порядке немецкой мнемоники (SIMATIC)

Немецкая мнемоника	Английская мнемоника	Каталог программных элементов	Описание
---| |---	---| |---	Bit logic Instruction	Нормально открытый контакт (адрес)
---|/|---	---|/|---	Bit logic Instruction	Нормально замкнутый контакт (адрес)
---( )	---( )	Bit logic Instruction	Выходная катушка
---(#)---	---(#)---	Bit logic Instruction	Промежуточный результат (Коннектор)
==0 ---| |---	==0 ---| |---	Status bits	Результат опроса на “равно 0”
>0 ---| |---	>0 ---| |---	Status bits	Результат опроса на “больше 0”
>=0 ---| |---	>=0 ---| |---	Status bits	Результат опроса на “больше или равно 0”
<=0 ---| |---	<=0 ---| |---	Status bits	Результат опроса на “меньше или равно 0”
<0 ---| |---	<0 ---| |---	Status bits	Результат опроса на “меньше 0”
<>0 ---| |---	<>0 ---| |---	Status bits	Результат опроса на “не равно 0”
ABS	ABS	Floating point Instruction	Вычисление абсолютной величины числа с плавающей точкой
ACOS	ACOS	Floating point Instruction	Вычисление арккосинуса угла для числа с плавающей точкой
ADD_DI	ADD_DI	Integer Math Instruction	Сложение двойных целых чисел
ADD_I	ADD_I	Integer Math Instruction	Сложение целых чисел
ADD_R	ADD_R	Floating point Instruction	Сложение чисел с плавающей точкой
ASIN	ASIN	Floating point Instruction	Вычисление арксинуса угла для числа с плавающей точкой
ATAN	ATAN	Floating point Instruction	Вычисление арктангенса угла для числа с плавающей точкой
BCD_DI	BCD_DI	Convert	Преобразовать двоично-десятичное число в двойное целое
BCD_I	BCD_I	Convert	Преобразовать двоично-десятичное число в целое
BIE ---| |---	BR ---| |---	Status bits	Бит ошибки "Регистр BR"
----(CALL)	----(CALL)	Program control	Вызов FC или SFC (без параметров)
CALL_FB	CALL_FB	Program control	Вызвать FB из списка блоков
CALL_FC	CALL_FC	Program control	Вызвать FC из списка блоков
CALL_SFB	CALL_SFB	Program control	Вызвать SFB из списка блоков
CALL_SFC	CALL_SFC	Program control	Вызвать SFC из списка блоков
CEIL	CEIL	Convert	Округлить до ближайшего большего целого числа
CMP >=D	CMP >=D	Compare	Сравнение двойных целых чисел (==, <>, >, <, >=, <=)
CMP >=I	CMP >=I	Compare	Сравнение целых чисел (==, <>, >, <, >=, <=)
CMP >=R	CMP >=R	Compare	Сравнение чисел с плавающей точкой (==, <>, >, <, >=, <=)

Немецкая мнемоника	**Английская мнемоника**	**Каталог программных элементов**	**Описание**
COS	COS	Floating point Instruction	Вычисление косинуса угла для числа с плавающей точкой
DI_BCD	DI_BCD	Convert	Преобразовать двойное целое число в двоично-десятичное
DI_R	DI_R	Convert	Преобразовать двойное целое число в число с плавающей точкой
DIV_DI	DIV_DI	Integer Math Instruction	Деление двойных целых чисел
DIV_I	DIV_I	Integer Math Instruction	Деление целых чисел
DIV_R	DIV_R	Floating point Instruction	Деление чисел с плавающей точкой
EXP	EXP	Floating point Instruction	Вычисление экспоненты
FLOOR	FLOOR	Convert	Округлить до ближайшего меньшего целого числа
I_BCD	I_BCD	Convert	Преобразовать целое число в двоично-десятичное
I_DI	I_DI	Convert	Преобразовать целое число в двойное целое
INV_I	INV_I	Convert	Инверсия целого числа
INV_DI	INV_DI	Convert	Инверсия двойного целого числа
---(JMP)	---(JMP)	Jumps	Безусловный переход
---(JMP)	---(JMP)	Jumps	Условный переход
---(JMPN)	---(JMPN)	Jumps	Переход по нулевому результату
LABEL	LABEL	Jumps	Метка
LN	LN	Floating point Instruction	Вычисление натурального логарифма
---(MCR>)	---(MCR>)	Program control	Выключить главное управляющее реле
---(MCR<)	---(MCR<)	Program control	Включить главное управляющее реле
---(MCRA)	---(MCRA)	Program control	Активировать главное управляющее реле
---(MCRD)	---(MCRD)	Program control	Деактивировать главное управляющее реле
MOD_DI	MOD_DI	Integer Math Instruction	Выделение остатка от деления двойных целых чисел
MOVE	MOVE	Move	Передача значения
MUL_DI	MUL_DI	Integer Math Instruction	Умножение двойных целых чисел
MUL_I	MUL_I	Integer Math Instruction	Умножение целых чисел
MUL_R	MUL_R	Floating point Instruction	Умножение чисел с плавающей точкой
---(N)---	---(N)---	Bit logic Instruction	Выделение отрицательного фронта RLO
NEG	NEG	Bit logic Instruction	Выделение отрицательного фронта сигнала
NEG_DI	NEG_DI	Convert	Дополнительный код двойного целого числа
NEG_I	NEG_I	Convert	Дополнительный код целого числа
NEG_R	NEG_R	Convert	Изменение знака числа с плавающей точкой

Немецкая мнемоника	**Английская мнемоника**	**Каталог программных элементов**	**Описание**
---| NOT |---	---| NOT |---	Bit logic Instruction	Инвертировать результат логической операции (поток энергии)
---(OPN)	---(OPN)	DB call	Открыть блок данных : DB или DI
OS ---| |---	OS ---| |---	Status bits	Бит ошибки переполнение с памятью
OV ---| |---	OV ---| |---	Status bits	Бит ошибки переполнение
---(P)---	---(P)---	Bit logic instructio	Выделение положительного фронта RLO
POS	POS	Bit logic Instruction	Выделение положительного фронта сигнала
---(R)	---(R)	Bit logic Instruction	Сбросить выход в 0
---(RET)	---(RET)	Program control	Возврат
ROL_DW	ROL_DW	Shift/Rotate	Циклический сдвиг двойного целого числа влево
ROR_DW	ROR_DW	Shift/Rotate	Циклический сдвиг двойного целого числа вправо
ROUND	ROUND	Convert	Округлить до двойного целого числа
RS	RS	Bit logic Instruction	RS -триггер
---(S)	---(S)	Bit logic Instruction	Установить выход в 1
---(SA)	---(SF)	Timers	Катушка таймера задержки выключения
---(SAVE)	---(SAVE)	Bit logic Instruction	Сохранение бита RLO в BR бите
S_AVERZ	S_OFFDT	Timers	S5 таймер “Задержка выключения”
---(SE)	---(SD)	Timers	Катушка таймера задержки включения
S_EVERZ	S_ODT	Timers	S5 таймер “Задержка включения”
SHL_DW	SHL_DW	Shift/Rotate	Сдвиг двойного слова влево
SHL_W	SHL_W	Shift/Rotate	Сдвиг слова влево
SHR_DI	SHR_DI	Shift/Rotate	Сдвиг двойного целого слова вправо
SHR_DW	SHR_DW	Shift/Rotate	Сдвиг двойного слова вправо
SHR_I	SHR_I	Shift/Rotate	Сдвиг целого слова вправо
SHR_W	SHR_W	Shift/Rotate	Сдвиг слова вправо
---(SI)	---(SP)	Timers	Катушка таймера “ Импульс”
S_IMPULS	S_PULSE	Timers	S5 таймер “ Импульс”
SIN	SIN	Floating point Instruction	Вычисление синуса угла числа с плавающей точкой
SQR	SQR	Floating point Instruction	Вычисление квадрата числа с плавающей точкой
SQRT	SQRT	Floating point Instruction	Вычисление квадратного корня числа с плавающей точкой
SR	SR	Bit logic Instruction	SR- триггер
---(SS)	---(SS)	Timers	Катушка таймера задержки включения с памятью
S_SEVERZ	S_ODTS	Timers	S5 таймер задержки включения с памятью
SUB_DI	SUB_DI	Integer Math Instruction	Вычитание двойных целых чисел
SUB_I	SUB_I	Integer Math Instruction	Вычитание целых чисел
SUB_R	SUB_R	Floating point Instruction	Вычитание чисел с плавающей точкой

Немецкая мнемоника	**Английская мнемоника**	**Каталог программных элементов**	**Описание**
---(SV)	---(SE)	Timers	Катушка таймер задержки включения
S_VIMP	S_PEXT	Timers	S5 таймер “Удлиненный импульс”
---(SZ)	---(SC)	Counters	Установка значения счетчика
TAN	TAN	Floating point Instruction	Вычисление тангенса числа с плавающей точкой
TRUNC	TRUNC	Convert	Выделение целой части числа
UO ---| |---	UO ---| |---	Status bits	Бит ошибки недопустимой операции
WAND_DW	WAND_DW	Word logic Instruction	Поразрядное И над двойными словами
WAND_W	WAND_W	Word logic Instruction	Поразрядное И над словами
WOR_DW	WOR_DW	Word logic Instruction	Поразрядное ИЛИ над двойными словами
WOR_W	WOR_W	Word logic Instruction	Поразрядное ИЛИ над словами
WXOR_DW	WXOR_DW	Word logic Instruction	Поразрядное исключающее ИЛИ над двойными словами
WXOR_W	WXOR_W	Word logic Instruction	Поразрядное исключающее ИЛИ над словами
ZAEHLER	S_CUD	Counters	Счетчик прямого и обратного счета
----(ZR)	----(CD)	Counters	Катушка счетчика обратного счета
Z_RUECK	S_CD	Counters	Счетчик обратного счета
---(ZV)	----(CU)	Counters	Катушка счетчика прямого счета
Z_VORW	S_CU	Counters	Счетчик прямого счета

B Примеры программирования

B.1 Обзор примеров программирования

Практические применения

Каждая команда KOP, описанная в данном руководстве выполняет определенную функцию. Комбинируя эти команды в программе, Вы можете реализовать широкий спектр задач автоматизации. Эта глава предлагает следующие примеры применения команд KOP:

- Управление транспортером с использованием битовой логики
- Определение направления движения ленты транспортера с использованием битовых логических операций
- Генерация тактовых импульсов с помощью таймерных команд
- Контроль склада с помощью операций счета и сравнения
- Решение задачи с использованием инструкций над целыми числами
- Установка интервала времени для нагревания печи

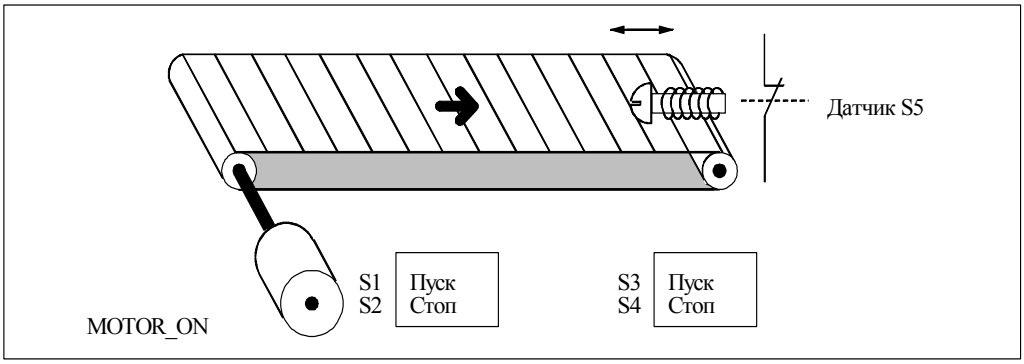
Используемые инструкции

Мнемоника	Каталог программных элементов	Описание
WAND_W	Word logic instruction	Поразрядное И со словами
WOR_W	Word logic instruction	Поразрядное ИЛИ со словами
--- (CD)	Counters	Обратный счет
--- (CU)	Counters	Прямой счет
---(R)	Bit logic instruction	Сброс выхода
---(S)	Bit logic instruction	Установка выхода
---(P)	Bit logic instruction	Обнаружение «+» фронта RLO
ADD_I	Floating-Point instruction	Сложение целых чисел
DIV_I	Floating-Point instruction	Деление целых чисел
MUL_I	Floating-Point instruction	Умножение целых чисел
CMP <=I, CMP >=I	Compare	Сравнение целых чисел
—| |—	Bit logic instruction	Нормально открытый контакт
—| / |—	Bit logic instruction	Нормально замкнутый контакт
—( )	Bit logic instruction	Выходная катушка
---(JMPN)	Jumps	Переход по нулевому значению
---(RET)	Program control	Возврат
MOVE	Move	Передача значения
--- (SE)	Timers	Таймер импульс с памятью

B.2 Пример: Битовые логические инструкции

Пример 1: Управление лентой транспортера

На следующем рисунке показана лента транспортера, которая может приводиться в движение с помощью электродвигателя. В начале транспортера имеются две кнопки: S1 для запуска и S2 для останова. В конце транспортера тоже имеются две кнопки: S3 для запуска и S4 для останова. Транспортер можно запускать или останавливать с любого конца. Также датчик S5 останавливает транспортер, когда предмет, находящийся на ленте, достигает конца.



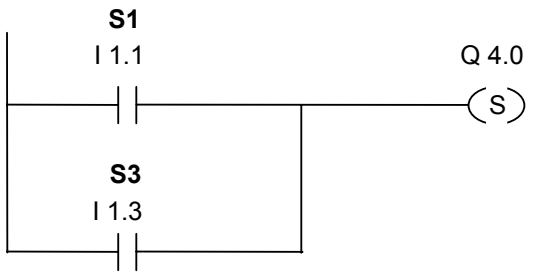
Абсолютное и символьное программирование

Вы можете написать программу для управления лентой транспортера, показанного на рисунке, используя **абсолютные значения** или их **символьные имена**, представляющие различные компоненты конвейера . Вы должны создать таблицу символов для того, чтобы поставить в соответствие выбранным символьным именам абсолютные адреса (см. STEP 7 Online Help).

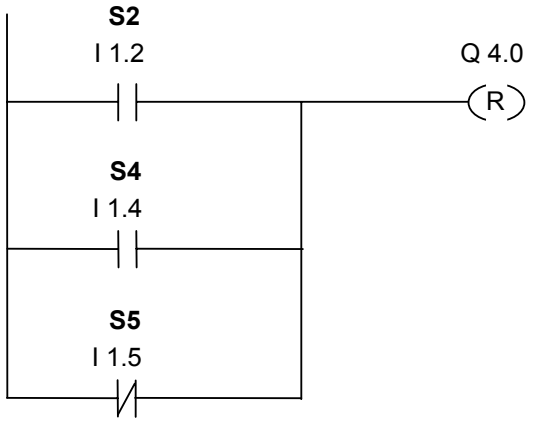
Компонент системы	**Абсолютный адрес**	**Символ**	**Таблица символов**
Кнопка “Пуск”	I 1.1	S1	I 1.1 S1
Кнопка “Стоп”	I 1.2	S2	I 1.2 S2
Кнопка “Пуск”	I 1.3	S3	I 1.3 S3
Кнопка “Стоп”	I 1.4	S4	I 1.4 S4
Датчик	I 1.5	S5	I 1.5 S5
Двигатель	Q 4.0	MOTOR_ON	Q 4.0 MOTOR_ON

Программа управления конвейером (в контактном плане)

Network 1: Нажатие любой кнопки “Пуск” включает двигатель.

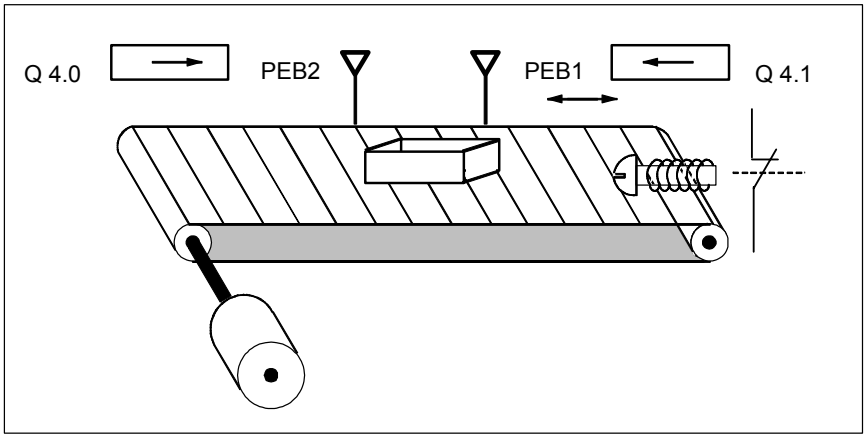


Network 2: Нажатие любой кнопки “Стоп” или срабатывание датчика вызывает отключение двигателя.



Пример 2 : Определение направления движения ленты транспортера

На рисунке показана лента транспортера, с двумя фотоэлектрическими датчиками (PEB1 и PEB2), которые служат для определения направления в котором движется пакет, расположенный на ленте. Оба фотобарьера работают как нормально открытые контакты



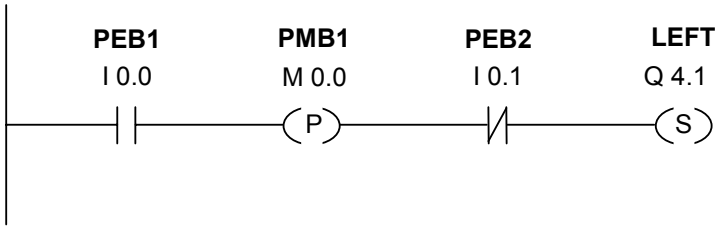
Абсолютное и символьное программирование

Вы можете написать программу для определения направления движения транспортера, показанного на рисунке, используя **абсолютные значения** или их **символьные имена**, представляющие различные компоненты конвейера . Вы должны создать таблицу символов для того, чтобы поставить в соответствие выбранным символьным именам абсолютные адреса (см. STEP 7 Online Help).

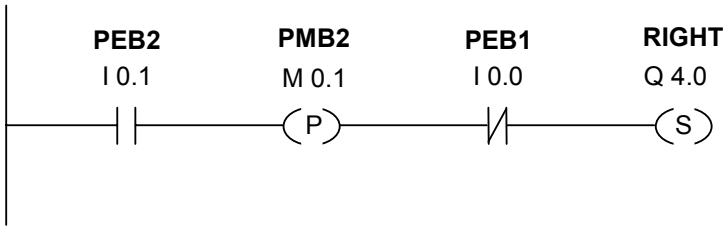
Компонент системы	**Абсолютный адрес**	**Символ**	**Таблица символов**
Фотоэлектрический датчик 1	I 0.0	PEB1	I 0.0 PEB1
Фотоэлектрический датчик 2	I 0.1	PEB2	I 0.1 PEB2
Индикатор движения направо	Q 4.0	RIGHT	Q 4.0 RIGHT
Индикатор движения налево	Q 4.1	LEFT	Q 4.1 LEFT
Тактовый меркер 1	M 0.0	PMB1	M 0.0 PMB1
Тактовый меркер 2	M 0.1	PMB2	M 0.1 PMB2

Программа для определения направления движения транспортера

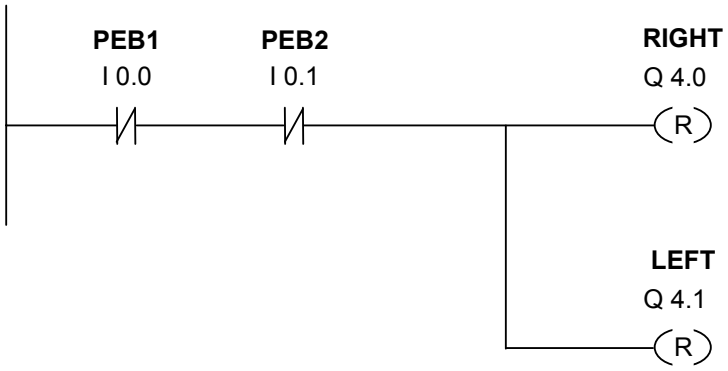
Network 1: Если на входе I0.0 сигнал изменяется с 0 на 1 (нарастающий фронт), и при этом состояние сигнала на входе I0.1 равно 0, то пакет на ленте транспортера движется влево.



Network 2: Если на входе I0.1 сигнал изменяется с 0 на 1 (нарастающий фронт), и при этом состояние сигнала на входе I0.0 равно 0, то пакет на ленте транспортера движется вправо. Если световой барьер выдает низкий уровень, это значит , что пакет находится перед ним.



Network 3: Если оба световых барьера перекрытыэто значит , что пакет находится между ними. Указатели направления сбрасываются.



B.3 Пример: Таймерные инструкции

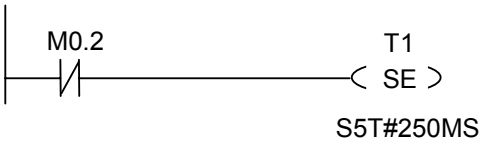
Генератор тактовых импульсов

Для создания периодически повторяющегося сигнала Вы можете использовать генератор тактовых импульсов или импульсное реле. Генераторы тактовых импульсов обычно используются в системах сигнализации, управляющих миганием индикаторных ламп.

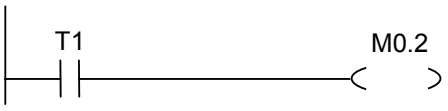
Если Вы используете S7-300, то Вы можете реализовать функцию генератора тактовых импульсов используя вызов программы из специальных организационных блоков, управляемых временем. Пример, показанный в следующей программе KOP, иллюстрирует использование таймерных функций для генерации тактовых импульсов.

Программа для синхронного генератора импульсов (в контактном плане)

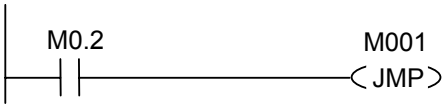
Network 1: Если состояние сигнала таймера T 1 равно 0, загрузить значение времени 250 мс в T 1 и запустить T 1 как удлинненный импульс.



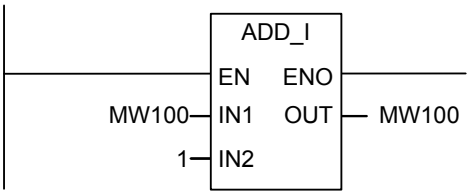
Network 2: Состояние таймера временно сохраняется во вспомогательном меркерном.



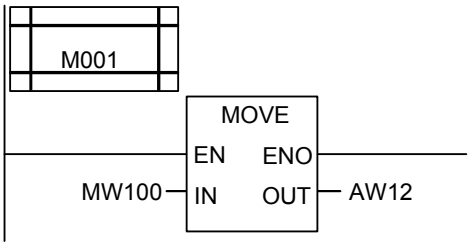
Network 3: Если состояние сигнала таймера T1 равно "1", перейти на метку M001.



Network 4: Когда время таймера T1 истекает, MW 100 увеличивается на "1".

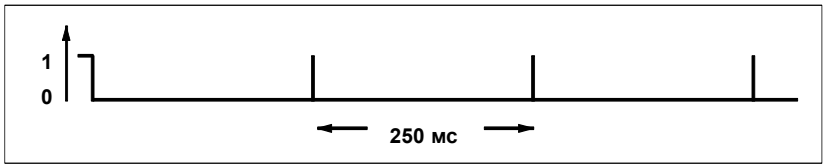


Network 5: Команда MOVE позволяет вывести различные тактовые частоты на выходах от Q12.0 до Q 13.7.



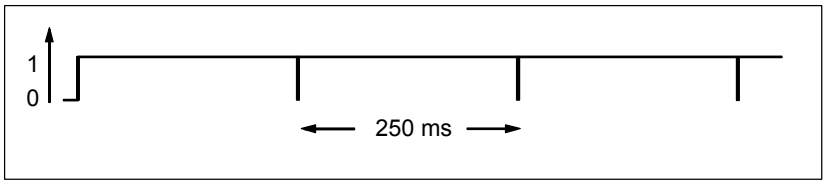
Проверка сигнала

Проверка значения T1 дает следующий результат логической операции(RLO) для нормально замкнутого контакта M0.2.



Как только время таймера истекает, таймер запускается вновь. Поэтому опрос сигнала, который выполняется командой --| / |-- T1, выдает состояние сигнала "1" только кратковременно.

Соответственно инвертированный RLO имеет вид:



Каждые 250 мс бит RLO становится равным 0. Переход не метку не выполняется и содержимое слова памяти MW100 увеличивается на 1.

Получение необходимой частоты

В таблице B–5 перечислены частоты, которые Вы можете получить из отдельных битов байтов памяти MB101 и MB100:

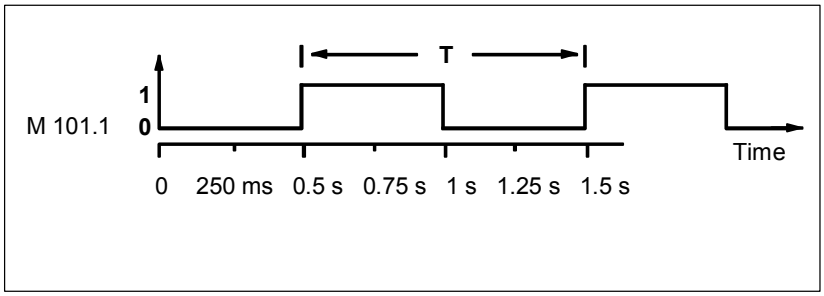
Биты MB101/MB100	Частота в Гц	Длительность
M 101.0	2.0	0.5 с (250 мс вкл./250 мс выкл.)
M 101.1	1.0	1 с (0.5 с вкл./0.5 с выкл.)
M 101.2	0.5	2 с (1 с вкл./1 с выкл.)
M 101.3	0.25	4 с (2 с вкл./2 с выкл.)
M 101.4	0.125	8 с (4 с вкл./4 с выкл.)
M 101.5	0.0625	16 с (8 с вкл./8 с выкл.)
M 101.6	0.03125	32 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.)

Состояние отдельных битов меркерного байта MB 101

Цикл	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Время в ms
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

Состояние сигнала бита 1 меркерного байта MB 101 (M 101.1)

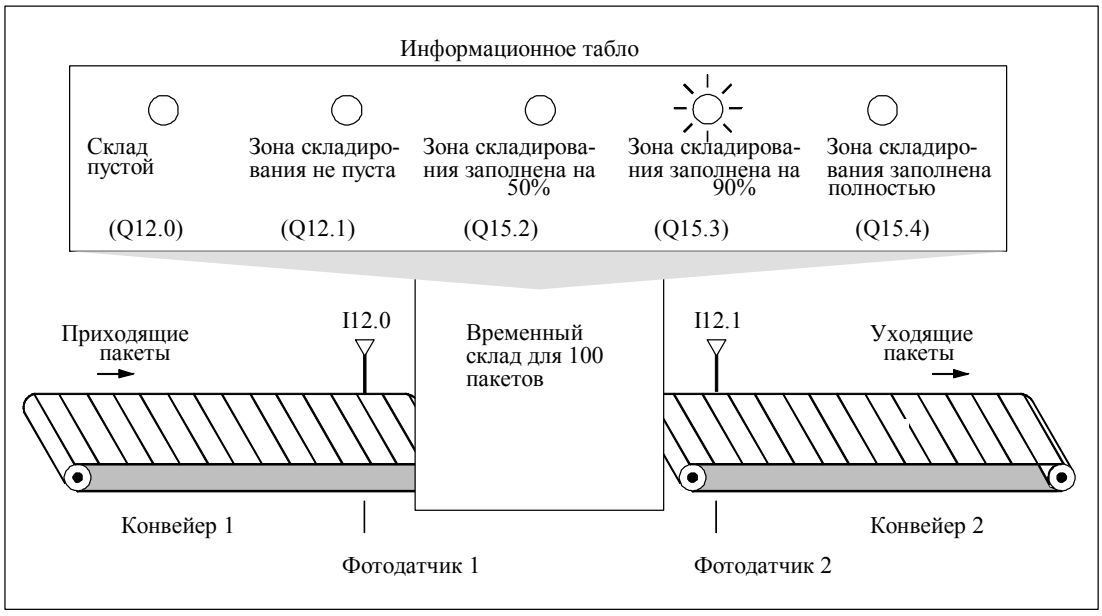
Частота = 1/T = 1/1 s = 1 Hz



B.4 Пример: Операции счета и сравнения

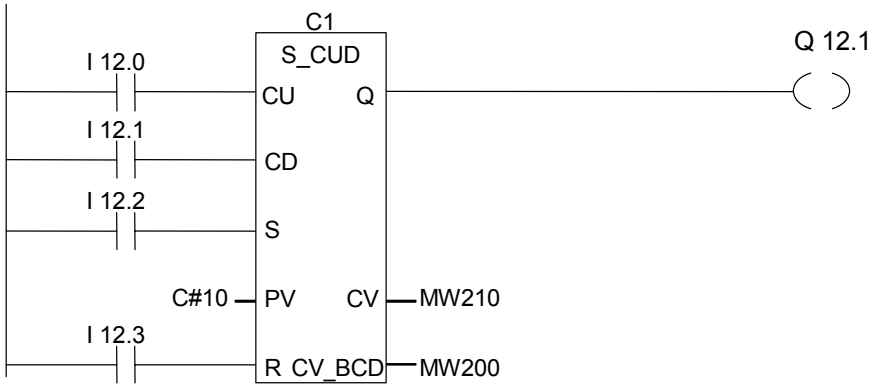
Промежуточный склад со счетчиком и компаратором

На рисунке представлена система из двух конвейеров и промежуточным складом между ними. Конвейер 1 доставляет пакеты в зону складирования. Фотоэлектрический датчик в конце конвейера 1 у зоны складирования определяет, сколько пакетов доставлено в эту зону. Конвейер 2 транспортирует пакеты из зоны временного складирования к погрузочной площадке, где пакеты грузятся на грузовые автомобили для доставки клиентам. Фотоэлектрический датчик в конце конвейера 2 вблизи зоны складирования определяет, сколько пакетов покидают зону складирования для отправки на погрузочную площадку. Информационное табло с пятью лампами отображает уровень заполнения склада временного хранения.



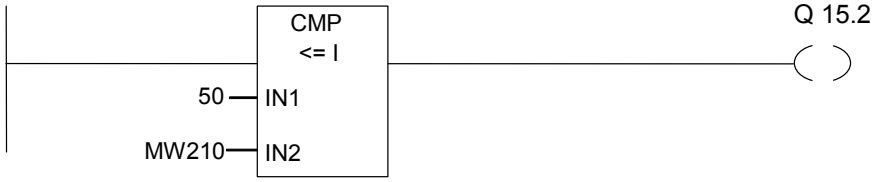
Программа для индикации загрузки склада (в контактном плане)

Network 1: Счетчик C1 увеличивает значение при каждом изменении сигнала с 0 на 1 на входе CU и уменьшает значение при каждом изменении сигнала с 0 на 1 на входе CD. При изменении сиг нала с 0 на 1 на входе S счетчик принимает значение PV. Изменение сигнала с 0 на 1 на входе R сбрасывает значение счетчика в 0. MW200 содержит текущее значение C1. Q12.1 показывает, что зона складирования не пуста.

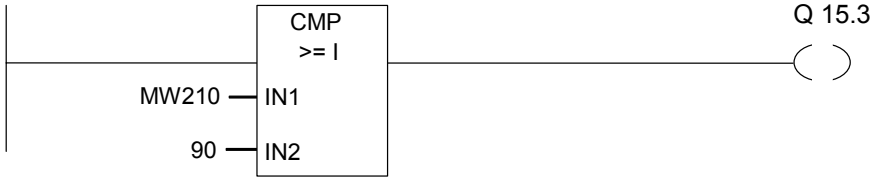


Network 2: Q12.0 показывает, что склад пуст.

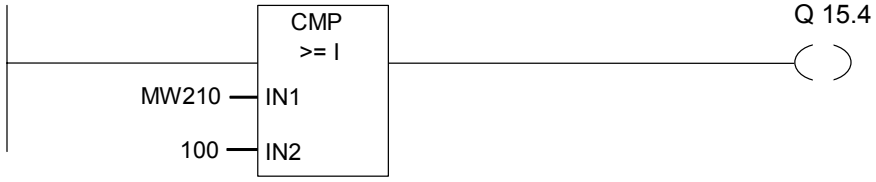
Network 3: Если 50 меньше или равно значению счетчика (иными словами, если текущее значение счетчика больше 50), то загорается индикаторная лампа “Зона складирования заполнена на 50%”



Network 4: Если значение счетчика больше или равно 90, то горит индикаторная лампа “Зона складирования заполнена на 90%”.



Network 5: Если значение счетчика больше или равно 100, то горит индикаторная лампа “Зона складирования заполнена полностью” .



B.5 Пример: Математические операции с целыми числами

Решение математической задачи

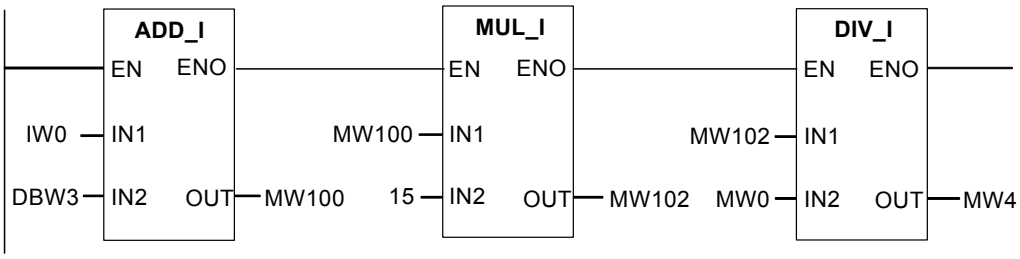
Следующий пример программы показывает, как использовать математические операции с целыми числами и команды L и T для вычисления результата следующего уравнения:

MW4 = ((IW0 + DBW3) x 15) / MW0

Программа в контактном плане

Network 1: Открыть блок данных DB1.

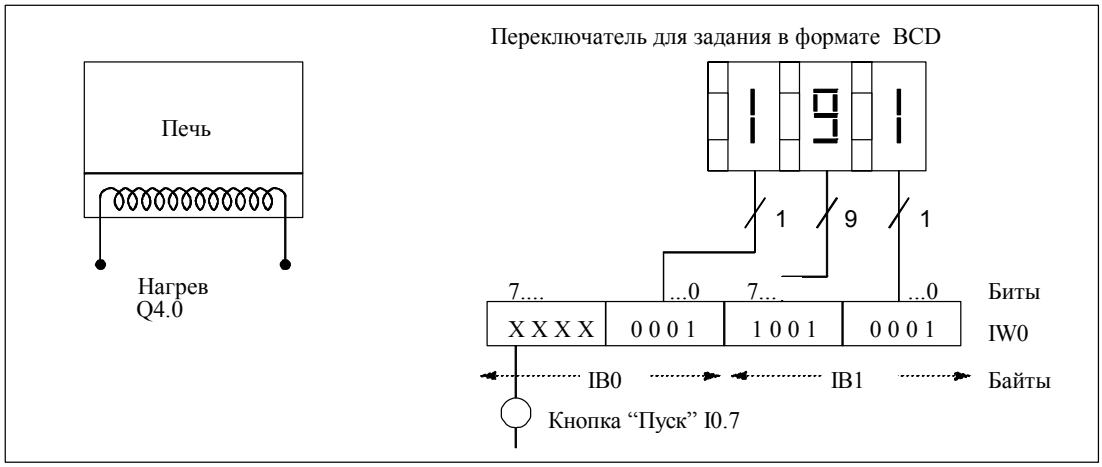
Network 2: Входное слово IW0 складывается с глобальным словом данных DBW3 (блок данных должен быть создан и открыт) и сумма загружается в меркерное слово MW100. MW100 затем умножается на 15 и результат сохраняется в меркерном слове MW102. MW102 делится на MW0. Результат сохраняется в MW4.



B.6 Пример: Поразрядные логические операции со словами

Нагрев печи

Оператор начинает разогрев печи нажатием кнопки “Пуск”. Он может задавать длительность нагрева с помощью цифрового переключателя, показанного на рисунке. Значение, устанавливаемое оператором задает количество секунд в двоично-десятичном формате (BCD).



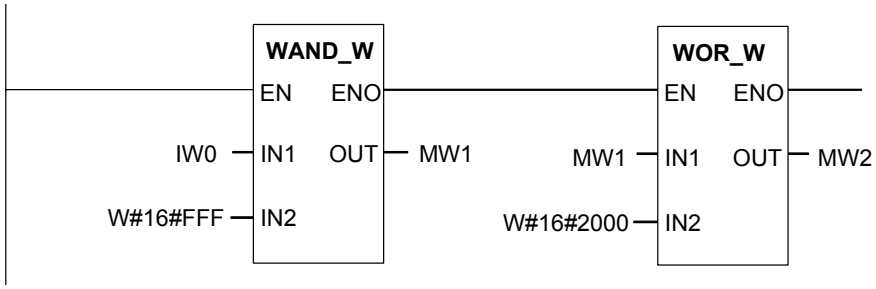
Компонент системы	**Абсолютный адрес в программе KOP**
Кнопка “Пуск”	I0.7
Переключатель для разряда единиц	от I1.0 до I1.3
Переключатель для разряда десятков	от I1.4 до I1.7
Переключатель для разряда сотен	от I0.0 до I0.3
Запуск нагрева	Q4.0

Программа управления нагревом печи в контактном плане

Network 1: Если таймер работает, то включается нагрев.

Network 2: Если таймер работает, инструкция **Return(Возврат)** заканчивает выполнение программы.

Network 3: Замаскировать биты с I0.4 по I0.7, т.е. сбросить их в 0. Эти биты входов переключателя не используются. 16 битов входов переключателя сопрягаются с W#16#0FFF в соответствии с командой “**Поразрядное И со словами**”. Результат загружается в меркерное слово MW1. Чтобы установить базу 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.



Network 4: Запуск таймера T 1 как удлинненный импульс при нажатии на стартовую кнопку, с установкой его на значение из MW2 (подготовленного ранее в сегменте 3).

C Работа в контактном плане

C.1 EN/ENO механизм

Деблокировка входа (EN) и разрешение на выходе (ENO) в контактном плане управляются при помощи бита BR.

Если EN и ENO определены, то выполняется следующее:

ENO = EN AND NOT (ошибка в блоке)

Если не происходит ошибки (ошибка в блоке = 0), то ENO = EN.

Механизм EN/ENO используется для:

- Математических инструкций,
- Инструкций передачи и преобразования,
- Инструкций сдвига и циклического сдвига,
- Вызова блоков.

Этот механизм не используется для:

- Инструкций сравнения,
- Счетчиков,
- Таймеров.

В зависимости от предшествующих вызову блока и последующих логических инструкций, кроме необходимых для работы инструкций, генерируются дополнительные STL инструкции для механизма EN/ENO. Четыре возможных варианта показаны для примера сложения:

1. Сложение с использованием заданий как EN так и ENO ,
2. Сложение с использованием задания EN и без задания ENO ,
3. Сложение без задания EN , но с заданием ENO ,
4. Сложение без задания EN и без задания ENO.

Замечания по созданию пользовательских блоков

При создании блоков для использования их в FBD или KOP, Вы должны обеспечить при выходе из блока установку в 1 бита BR. Следующие четыре примера показывают как это получается делается. Вы не можете использовать бит BR как меркер, потому что он постоянно переписывается механизмом EN/ENO . Для сохранения сообщения об ошибке, используйте временные переменные . Сбросьте ее значение в 0 в начале выполнения программы. В каждой точке программы, в которой Вы предполагаете, что произошла ошибка, Вы можете установить этот бит в “1” и вывести сообщение об ошибочной работе блока, используя механизм EN/ENO . При этом могут использоваться инструкции A NOT и SET. Пример заключительного сегмента программы:

```
end:    AN error
        SAVE
```

Убедитесь, что эта часть программы будет обработана при любых условиях, это означает недопустимость использования инструкции BEC в этом блоке и пропуск этой инструкции командами перехода.

C.1.1 Сложение с заданием EN и ENO

При сложении с использованием EN и ENO заданий, следующие STL инструкции будут созданы:

```
1  A   I   0.0          // Задание входа EN
2  JNB _001             // Запись RLO в BR и переход при RLO = 0
3  L     in1           // Загрузка слагаемого 1
4  L     in2           // Загрузка слагаемого 2
5         +I           // Сложение
6  T     out           // Вывод значения суммы
7  AN    OV            // Оценка переполнения
8  SAVE                // Сохранение ошибки в BR
9  CLR                 // Первичный опрос
10 _001: A     BR      // Запись BR в RLO
11 =     Q   4.0
```

После выполнения первой инструкции формируется результат логической операции. Инструкция JNB записывает RLO в бит BR и взводит бит первичного опроса.

- Если RLO = 0, выполняется переход на метку (строка 10) и выполняется инструкция опроса бита BR. Сложение не выполняется. В строке 10 значение бита BR копируется в RLO и тогда, соответственно, значение 0 передается на выход.
- При RLO = 1, переход на метку не выполняется и, соответственно , выполняется сложение. В строке 7 оценивается возможная ошибка при сложении, и затем, результат этого сохраняется в бите BR в строке 8. Строка 9 взводит бит первичного опроса. Затем бит BR записывается в RLO в строке 10 и таким образом выход показывает было ли успешно выполнено сложение. Строки 10 и 11 не меняют бит BR.

C.1.2 Сложение с использованием EN и без задания ENO

В случае сложения с заданным входом EN и без задания выхода ENO, формируется следующая STL программа:

```
1  A     I      0.0    // EN задание
2  JNB _001         // Запись RLO в BR и переход на метку при RLO = 0
3  L     in1           // Загрузка слагаемого 1
4  L     in2           // Загрузка слагаемого 2
5  +I                  // Сложение
6  T     out           // Передача результата на выход
7  _001:  NOP   0
```

После выполнения инструкции строки 1, RLO содержит результат выполненной операции. Инструкция JNB записывает RLO в бит BR и взводит бит первичного опроса.

- Если RLO = 0, выполняется переход на метку (строка 7). Сложение не выполняется. Биты BR и RLO принимают значение 0.
- При RLO = 1, переход на метку не выполняется и, соответственно , выполняется сложение. В программе не оценивается возможная ошибка сложения. Биты BR и RLO принимают значение 1.

C.1.3 Сложение без задания EN , но с использованием выхода ENO

В случае сложения без использованием входа EN, но с заданием выхода ENO, формируется следующая STL программа :

```
1  L     in1     // Загрузка слагаемого 1
2  L     in2     // Загрузка слагаемого 2
3  +I            // Сложение
4  T     out     // Передача результата на выход
5  AN    OV      // Оценка переполнения
6  SAVE          // Сохранение ошибки в BR
7  CLR           // Первичный опрос
8  A     BR      // Запись BR в RLO
9  =  Q   4.0
```

Сложение выполняется безусловно. В строке 5 программы оценивается возможная ошибка переполнения при сложении, которая затем сохраняется в бите BR в строке 6. Строка 7 взводит бит первичного опроса. Теперь бит BR записывается в бит RLO в строке 8 и этим , соответственно, показывает была ли ошибка при выполнении сложения или нет.

Строки 8 и 9 не изменяют содержания бита BR, показывающего успешное выполнение задачи.

C.1.3 Сложение без использованием входа EN и без задания ENO

В случае сложения без использованием входа EN и без задания выхода ENO, формируется следующая STL программа:

```
1  L     in1          // Загрузка слагаемого 1
2  L     in2          // Загрузка слагаемого 2
3  +I                 // Сложение
4  T     out          // Передача результата на выход
5  NOP 0
```

Сложение выполняется безусловно. Биты BR и RLO не изменяют своего значения.

C.2 Передача параметров

Формальные параметры блока должны получить для обработки актуальные значения. Функциональные блоки копируют эти значения в экземплярный блок данных, указанный при вызове функционального блока. При работе функций указатель на область фактического значения располагается в локальном стеке вызывающего блока. Перед вызовом блока значения входных параметров копируются в экземплярный блок DB или в L -стек. После окончания обработки блока выходные величины передаются в выходные переменные. Внутри вызываемого блока Вы работаете только с копией фактических параметров. Для этого в вызывающем блоке создаются необходимые STL инструкции, которые скрыты от пользователя.

Примечание

Если меркеры, входы, выходы или периферия используются в качестве фактических адресов функции , они обрабатываются несколько отлично от других адресов. При этом обновление производится напрямую, без L-стека.

Исключение: Если входной параметр имеет тип данных BOOL , то фактическое значение, передается в блок через L -стек.

Предупреждение:

При программировании вызываемого блока, убедитесь, что к параметрам, описанным как выходные обращение производится только на запись. В противном случае на выход может попасть случайная величина! В случае функциональных блоков, значение располагается в экземплярном блоке данных, указанном при последнем вызове.При работе с функцией, значение будет передаваться с использованием L -стека.

Учтите следующие моменты:

- Задайте начальные значения для всех выходных параметров, если это возможно.
- Старайтесь не использовать RLO –зависимые инструкции сброса и установки битов. Если RLO имеет значение 0, в результате может быть получена случайная величина.
- Если Вы программируете переходы внутри блока, убедитесь, что не будут исключены из обработки необходимые Вам инструкции записи выходных параметров. На забудьте об этом при использовании инструкций BEC и MCR .

Предметный указатель

Д

З

И

К

М

Н

О

П

C

У

Ц

A

B

C

D

E

I

K

L

M

N

O

P

R

S

U

W

T

X

Z