SIEMENS

SIMATIC

Программирование в функциональном плане (FBD) для S7-300 и S7-400

Справочное руководство

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

Редакция 01/2004
A5E00261409-01

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 FBD (функциональный план).

Кроме того, в этом руководстве имеется справочный раздел по элементам языка программирования FBD, 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 примениенения руководства

Это руководство разработано для программного пакета STEP 7 версии 5.3.

Соответствие стандартам

FBD соответствует языку ”Function Block Diagram” (“Функциональный план”), определенному в стандарте Международной электротехнической комиссии IEC 1131–3. По поводу дальнейших 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 (LAD)/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/LAD/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 Function Block Diagram". 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**t**s) или просто битами. После выполнения операций И, ИЛИ, исключающее ИЛИ значение «1» на выходе означает логическое ДА, а «0» - логическое НЕТ.

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

- И, ИЛИ и исключающее ИЛИ: эти инструкции опрашивают состояние сигнала и выдают результат, который или копируется в бит RLO, или комбинируются с ним.
- Логические инструкции И перед ИЛИ и ИЛИ перед И.
- Присваивание и коннектор: эти инструкции присваивают значение RLO или сохраняют его временно.

Выполнение следующих инструкций зависит от RLO = 1:

- S:Установить выход
- R:Сбросить выход
- SR, RS:Триггер S/R и триггер R/S

Некоторые инструкции реагируют на нарастающий или падающий фронт:

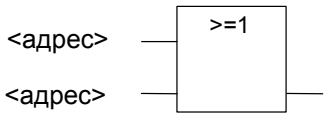
- N:Выделение отрицательного фронта RLO
- P: Выделение положительного фронта RLO
- NEG:Выделение отрицательного фронта сигнала
- POS:Выделение положительного фронта сигнала

Остальные инструкции непосредственно влияют на RLO следующим образом:

- Инвертируют RLO
- SAVE:Сохраняют RLO в бите двоичного результата слова состояния

1.2 >=1 : Логическая операция OR (ИЛИ)

Обозначение



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

Описание В случае инструкции **ИЛИ** Вы можете опросить состояния сигналов по двум или более адресам, указанным на входах блока **ИЛИ**.

Если состояние сигнала хотя бы по одному из адресов равно 1, то условие удовлетворяется и инструкция выдает результат 1. Если состояние сигнала по всем адресам равно 0, то условие не удовлетворяется и инструкция дает результат 0.

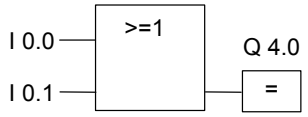
Если инструкция **ИЛИ** является первой в цепи логических операций, то она сохраняет результат опроса состояния сигнала в бите RLO.

Каждая инструкция **ИЛИ**, не являющаяся первой в цепи логических операций, комбинирует результат опроса состояния сигнала со значением, хранящимся в бите RLO. Эти значения комбинируются в соответствии с таблицей истинности для И

Биты слова состояния

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

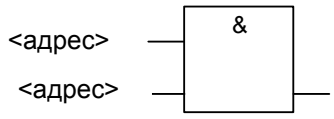
Пример



Выход Q4.0 установлен , когда равно 1 состояние сигнала на входе I0.0 ИЛИ на входе I0.1.

1.3 & : Логическая инструкция И

Обозначение



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

Описание

В случае инструкции **И,** Вы можете опросить состояния сигналов по двум или более адресам, указанным на входах блока И.

Если состояния сигналов по всем адресам равны 1, то условие удовлетворяется и операция выдает результат 1. Если состояние сигнала хотя бы по одному адресу равно 0, то условие не удовлетворяется и инструкция выдает результат 0.

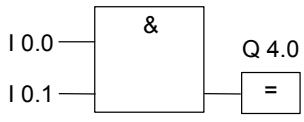
Если инструкция **И** является первой в цепи логических операций, то она сохраняет результат опроса состояния сигнала в бите RLO.

Каждая инструкция **И**, не являющаяся первой в цепи логических операций, комбинирует результат опроса состояния сигнала со значением, хранящимся в бите RLO. Эти значения комбинируются в соответствии с таблицей истинности для И.

Биты слова состояния

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

Пример



Выход Q4.0 установлен, когда равны 1 состояния сигналов на входах I0.0 **И** I0.1.

1.4 Логические инструкции И перед ИЛИ и ИЛИ перед И

Описание

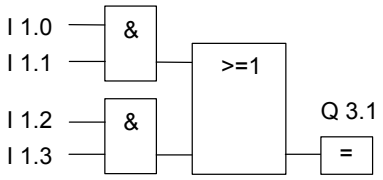
В случае инструкции **И-перед-ИЛИ** результат инструкции определяется в соответствии с таблицей истинности для ИЛИ.

В случае логической инструкции **И-перед-ИЛИ** состояние сигнала равно 1, если удовлетворяется хотя бы одна логическая операция И.

Биты слова состояния

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

Пример



Состояние сигнала на выходе Q3.1 равно 1, когда удовлетворяется хотя бы одна операция И.

Состояние сигнала на выходе Q3.1 равно 0, когда не удовлетворяется ни одна операция И.

Описание

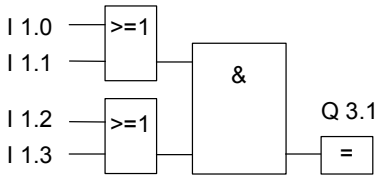
В случае инструкции **ИЛИ-перед-И** результат инструкции определяется в соответствии с таблицей истинности для И.

В случае логической инструкции **ИЛИ-перед-И** состояние сигнала равно 1, когда удовлетворяются все логические инструкции ИЛИ.

Биты слова состояния

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

Пример

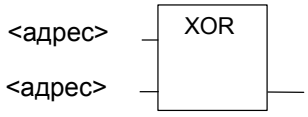


Состояние сигнала на выходе Q3.1 равно 1, когда удовлетворяются обе логические инструкции ИЛИ .

Состояние сигнала на выходе Q3.1 равно 0, когда по крайней мере одна из логических операций ИЛИ не удовлетворяется.

1.5 XOR : Логическая инструкция исключающее ИЛИ

Обозначение



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

Описание

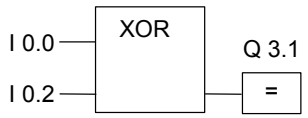
В случае инструкции **исключающее ИЛИ** результат инструкции определяется в соответствии с таблицей истинности для исключающего ИЛИ

В случае инструкции **исключающее ИЛИ** состояние сигнала равно 1, когда состояние сигнала одного из двух указанных адресов равно 1.

Биты слова состояния

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

Пример



Состояние сигнала на выходе Q3.1 равно 1когда равно 1 состояние сигнала ИЛИ на входе I0.0 .ИЛИ на входе I0.2 (но не на обоих одновременно).

1.6 Добавление двоичного входа

Обозначение

<адрес>

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

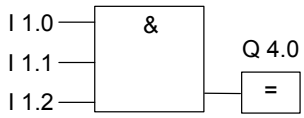
Описание

Операция **добавления двоичного входа** вставляет дополнительный вход в блок И, ИЛИ или Исключающее ИЛИ .

Биты слова состояния

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

Пример



Выход Q4.0 равен 1,когда равно 1 состояние всех входов : I1.0 И I1.1 И I1.2 .

1.7 Инверсия двоичного входа

Обозначение

Описание

Инструкция **Инверсия двоичного входа** инвертирует RLO.

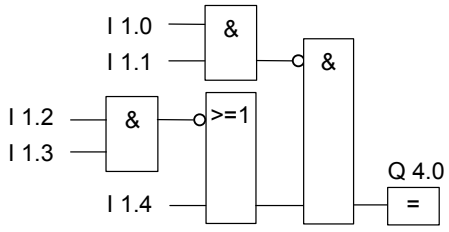
При отрицании результата логической инструкции Вы должны помнить следующие правила:

- Если инвертируется результат логической инструкции на первом входе блока И или ИЛИ, то логического сопряжения с RLO не производится.
- Если инвертируется результат логической инструкции не на первом входе блока ИЛИ, то вся логическая операция перед этим входом включается в логическую операцию ИЛИ.
- Если инвертируется результат логической инструкции не на первом входе блока И, то вся логическая операция перед этим входом включается в логическую операцию И.

Биты слова состояния

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

Пример



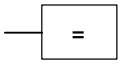
Выход Q4.0 равен 1 когда:

- состояния сигналов на I1.0 И I1.1 не равны одновременно 1
- И состояния сигналов на I1.2 И I1.3 не равны одновременно 1
- ИЛИ состояние сигнала на I1.4 не равно 1.

1.8 = : Инструкция присвоения

Обозначение

<адрес>



Параметры	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, D, L	Адрес указывает бит, которому присваивается значение результата цепи логических операций.

Описание

Инструкция **Присвоение** выдает результат логической инструкции. Этот блок в конце логической инструкции имеет состояние 1 или 0 в соответствии со следующими критериями:

- Сигнал на выходе равен 1, когда удовлетворяются условия логической инструкции перед выходным блоком
- Сигнал на выходе равен 0, когда условия логической инструкции перед выходным блоком не удовлетворяются.

Эта логическая операция FBD присваивает состояние сигнала выходу, указанному в качестве адреса. Если условия логической инструкции FBD удовлетворяются, то значение сигнала 1 присваивается адресу , указанному в инструкции. В противном случае состояние сигнала равно 0. На инструкцию **Присвоение** оказывает влияние главное управляющее реле (MCR).

За более подробной информацией о функциях MCR обратитесь к соответствующему разделу.

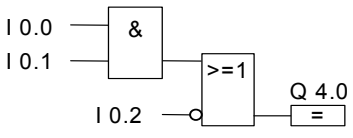
Блок **Присвоение** можно поместить только справа в конце цепочки логических операций. Однако, Вы можете использовать несколько таких блоков.

Вы можете создать инверсное присвоение с помощью инструкции **Инверсия двоичного входа.**

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 равно 1,когда: равны 1 состояния сигналов на входах I0.0 И I0.1, ИЛИ I0.2 равен 0.

1.9 #: Коннектор

Обозначение

<адрес>

Параметры	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, D, *L	Адрес указывает бит, которому присвоено значение RLO

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

Описание

Инструкция **Коннектор** - это промежуточный элемент, который запоминает RLO в буфере. Более точно, этот элемент буферизует битовую логическую операцию последней ветви, открытой перед коннектором.

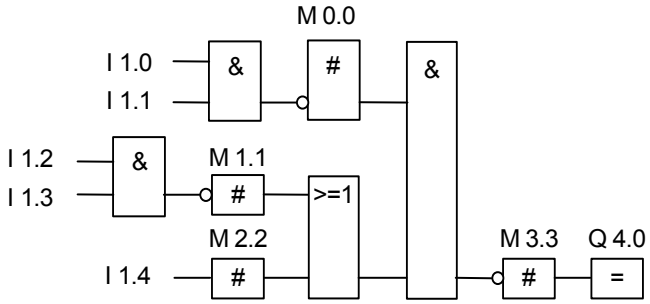
На инструкцию **Коннектор** оказывает влияние Главное управляющее реле (MCR). За более подробной информацией о функциях MCR обратитесь к разделу 20.4.

Вы можете создать инвертированный коннектор с помощью отрицания входа коннектора

Биты слова состояния

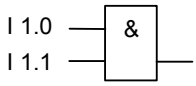
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	0	X	-	1

Пример

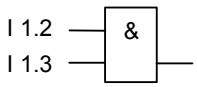


Коннекторы сохраняют следующие промежуточные результаты логических операций:

M0.0 производит промежуточное сохранение инверсного RLO для



M1.1 производит промежуточное сохранение инверсного RLO для



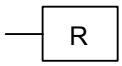
M2.2 производит промежуточное сохранение статуса I1.4

M3.3 сохраняет инверсный результат всей логической операции.

1.10 R: Сброс бита

Обозначение

<адрес>



Параметры	Тип данных	Область памяти	Описание
<адрес>	BOOL TIMER COUNTER	I, Q, M, T, C, D, L	Адрес указывает бит который будет сброшен

Описание

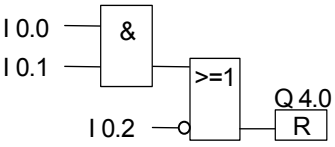
Инструкция **Сбросить бит** исполняется только тогда, когда RLO равен 1. Если RLO равен 1, эта инструкция сбрасывает указанный адрес в 0. Если RLO равен 0, то инструкция не влияет на указанный адрес, который остается неизменным.

На инструкцию **Сбросить бит** оказывает влияние Главное управляющее реле (MCR). За более подробной информацией о функциях MCR обратитесь к соответствующему разделу.

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 сбрасывается в 0 только тогда, когда:

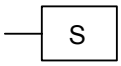
- Равны 1 состояния сигналов на входе I0.0 И I0.1
- ИЛИ состояние сигнала на входе I0.2 равно 0

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

1.11 S : Установка бита

Обозначение

<адрес>



Параметры	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, D, L	Адрес указывает, какой бит будет установлен.

Описание

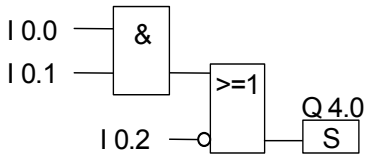
Инструкция **Установить бит** исполняется только тогда, когда RLO равен 1. Если RLO равен 1, эта инструкция устанавливает указанный адрес в 1. Если RLO равен 0, то инструкция не влияет на указанный адрес, который остается неизменным.

На инструкцию **Установить бит** оказывает влияние Главное управляющее реле (MCR). За более подробной информацией о функциях MCR обратитесь к соответствующему разделу.

Биты слова состояния

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

Пример



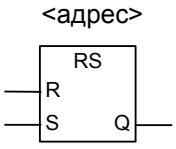
Состояние сигнала на выходе Q4.0 устанавливается в 1 только тогда, когда:

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

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

1.12 RS : RS- Триггер

Обозначение



Параметры	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, D, L	Адрес указывает, какой бит будет установлен или сброшен
S	BOOL	I, Q, M, D, L, T, C	Инструкция установки
R	BOOL	I, Q, M, D, L, T, C	Инструкция сброса
Q	BOOL	I, Q, M, D, L	Состояние сигнала <адрес>

Описание

Инструкция **Сбросить / установить триггер** выполняет установку (S) или сброс (R) только тогда, когда RLO = 1. RLO, равный 0, не оказывает влияния на эти инструкции, адрес, указанный в команде остается неизменным

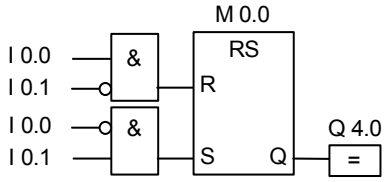
RS -триггер сбрасывается, когда состояние сигнала на входе R равно 1, а состояние сигнала на входе S равно 0. Если вход R равен 0, а вход S равен 1, то триггер установлен. Если RLO на обоих входах равно 1, то триггер установлен.

На инструкцию **Сбросить / установить триггер** оказывает влияние Главное управляющее реле (MCR). За более подробной информацией о функционировании MCR обращайтесь к соответствующему разделу.

Биты слова состояния

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

Пример



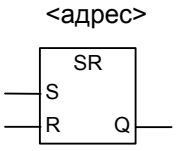
Если I0.0= 1, а I0.1 = 0, то меркер M0.0 сброшен и выход Q4.0 равен 0.
Если I0.0 = 0 а I0.1= 1, то меркер M0.0 установлен и выход Q4.0 = 1.

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.

1.13 SR : SR- Триггер

Обозначение



Параметры	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, D, L	Адрес указывает какой бит будет установлен или сброшен.
S	BOOL	I, Q, M, D, L, T, C	Инструкция установки
R	BOOL	I, Q, M, D, L, T, C	Инструкция сброса
Q	BOOL	I, Q, M, D, L	Состояние сигнала <адрес>

Описание

Инструкция **Установить / сбросить триггер** выполняет установку (S) или сброс (R) только тогда, когда RLO = 1. RLO, равный 0, не оказывает влияния на эти инструкции, адрес, указанный в команде остается неизменным.

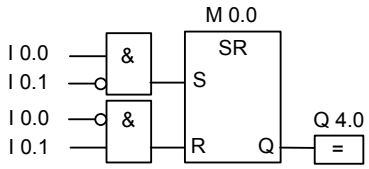
Триггер SR устанавливается, когда состояние сигнала на входе S равно 1, а на входе R равно 0. Если вход S равен 0, вход R равен 1, то триггер сбрасывается. Если RLO на обоих входах равно 1, то триггер сброшен.

На инструкцию **Установить / сбросить триггер** оказывает влияние Главное управляющее реле (MCR). За более подробной информацией о функционировании MCR обращайтесь к соответствующему разделу.

Биты слова состояния

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

Пример



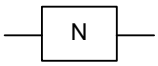
Если I0.0 = 1 а I0.1 = 0, то меркер M0.0 установлен и Q4.0 = 1.Если I0.0= 0 а I0.1 = 1, то меркер M0.0 сброшен и Q4.0 = 0.

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.

1.14 N: Выделение отрицательного фронта RLO

Обозначение

<адрес>



Параметры	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, D, L	Адрес указывает, какой меркер фронта будет хранить предыдущий RLO

Описание

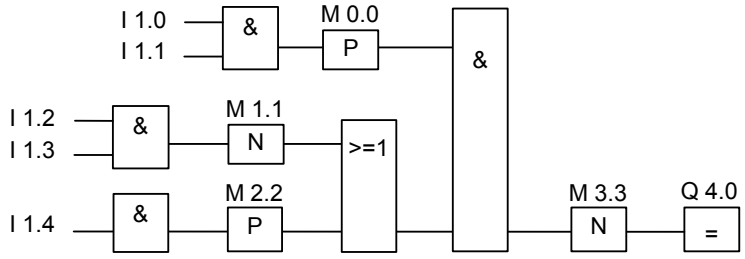
Инструкция **Выделение отрицательного фронта RLO** обнаруживает изменение с 1 на 0 (падающий фронт) по указанному адресу и отображает это установкой RLO в 1 после выполнения инструкции. Текущее состояние RLO сравнивается с состоянием сигнала операнда (меркер фронта). Если состояние сигнала операнда равно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

.

Биты слова состояния

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

Пример

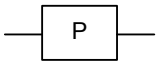


Меркер фронта M3.3 сохраняет состояние сигнала предыдущего RLO.

1.15 P : Выделение положительного фронта RLO

Обозначение

<адрес>



Параметры	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, D, L	Адрес указывает, какой меркер фронта будет хранить предыдущее RLO.

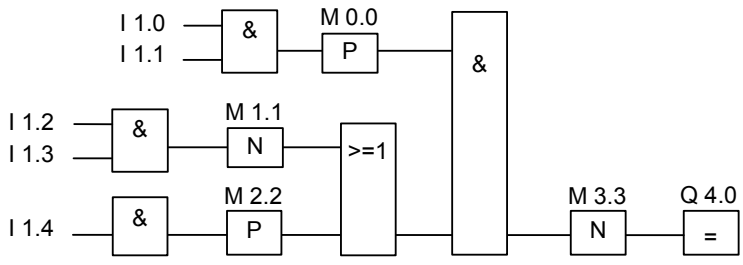
Описание

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 (меркер фронта). Если состояние сигнала операнда равно 0, а RLO равен 1 перед выполнением инструкции, то RLO будет равен 1 (импульс) после выполнения инструкции, во всех остальных случаях RLO равен 0. RLO перед инструкцией сохраняется в указанном операнде.

Бит слова состояния

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

Пример



Меркер фронта M3.3 сохраняет состояние сигнала предыдущего RLO.

1.16 SAVE : Сохранить RLO в бите BR

Обозначение

Описание

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.

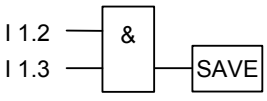
Не рекомендуется использовать команду **“Сохранить RLO в бите BR”** (LAD, FBD, STL), в следующих случаях , более подробно описанных в руководствах и встроенной справке: совместно с опросом бита BR в том же блоке, так как этот бит может быть изменен многими промежуточными инструкциями. Рекомендуется использовать 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 .

С помощью команды **“Сохранить RLO в бите BR”,** RLO некоторого сегмента может образовывать часть логической операции в подчиненном блоке. Команда CALL в вызывающем блоке сбрасывает бит первичного опроса.

Биты слова состояния

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

Пример

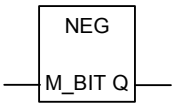


Результат логической операции (RLO) записывается в бит BR .

1.17 NEG: Выделение отрицательного фронта сигнала

Обозначение

<адрес>



Параметры	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, D, L	Сигнал, контролируемый на отрицательный (падающий) фронт.
M_BIT	BOOL	Q, M, D	Адрес M BIT указывает меркер, в котором хранится предыдущее состояние сигнала NEG. Для M BIT используйте область отображения входов I только если ни один из модулей не использует этот адрес.
Q	BOOL	I, Q, M, D, L	Импульсный выход.

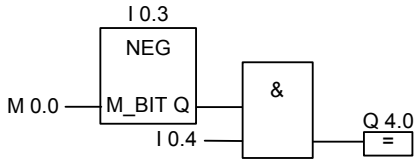
Описание

Инструкция **Выделение отрицательного фронта сигнала** сравнивает сигнала, который хранится в параметре M_BIT. Если происходит изменение с 1 на 0, то выход Q имеет значение 1, во всех остальных случаях он равен 0. Состояние сигнала с состоянием предыдущего опроса располагается в операнде <адрес> ..

Бит слова состояния

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

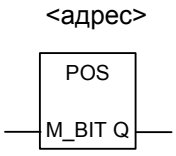
Пример



Выход Q4.0 = 1 когда имеет место падающий фронт на входе I0.3 И состояние сигнала на входе I0.4 равно 1.

1.18 POS: Выделение положительного фронта сигнала

Обозначение



Параметры	Тип данных	Область памяти	Описание
<адрес>	BOOL	I, Q, M, D, L	Сигнал ,контролируемый на положительный (нарастающий) фронт
M_BIT	BOOL	Q, M, D	Адрес M BIT указывает меркер, в котором хранится предыдущее состояние сигнала POS. Для M BIT используйте область отображения входов I только если ни один из модулей не использует этот адрес
Q	BOOL	I, Q, M, D, L	Импульсный выход

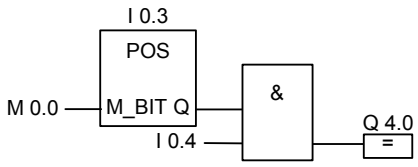
Описание

Инструкция **Выделение положительного фронта сигнала** сравнивает состояние сигнала в <адрес1> с предыдущим состоянием сигнала, который хранится в параметре M_BIT. Если происходит изменение с 0 на 1, то выход Q имеет значение 1, во всех остальных случаях он равен 0.

Бит слова состояния

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

Пример



Выход Q4.0 = 1 когда имеет место нарастающий фронт на входе I0.3 И состояние сигнала на входе I0.4 равно 1

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

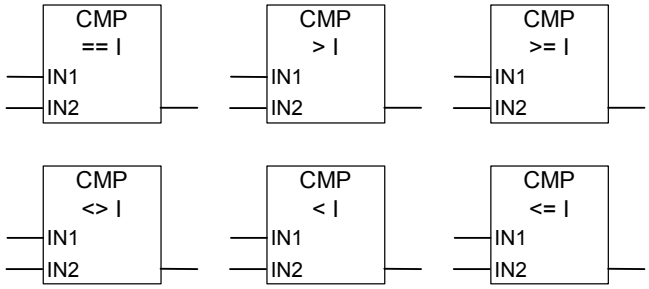
Если условие сравнения выполняется, то RLO получает значение "1". В противном случае RLO получает значение 0. Вы не можете инвертировать результат сравнения. Для этого возьмите другую инструкцию сравнения.

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
IN1	INT	I, Q, M, D, L или константа	Первое сравниваемое значение
IN2	INT	I, Q, M, D, L или константа	Второе сравниваемое значение
Выход блока	BOOL	I, Q, M, D, L	Результат сравнения

Описание

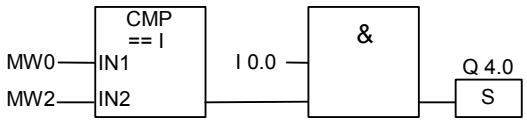
Инструкция **Сравнить целые числа** сравнивает значения двух 16-битных чисел с фиксированной точкой. Эта инструкция сравнивает входы IN1 и IN2 в соответствии с типом сравнения, выбираемым из окна списка.

.

Биты слова состояния

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

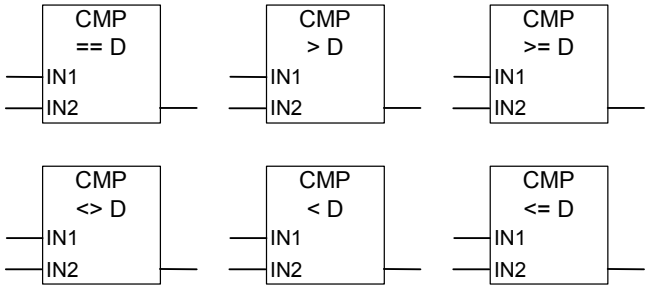
Пример



Q 4.0 устанавливается если:MW0 равно MW2 и состояниея сигнала на входе I0.0 равно 1.

2.3 CMP ? D : Сравнение чисел типа Double Integer

Обозначение



Параметр	Тип данных	Область памяти	Описание
IN1	DINT	I, Q, M, D, L или константа	Первое сравниваемое значение
IN2	DINT	I, Q, M, D L или константа	Второе сравниваемое значение
Выход блока	BOOL	I, Q, M, D, L	Результат сравнения

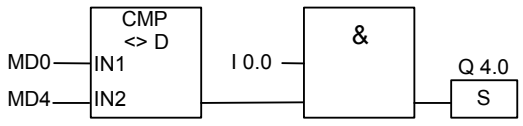
Описание

Инструкция **Сравнить двойные целые числа** сравнивает значения двух 32-битных чисел с фиксированной точкой. Эта инструкция сравнивает входы IN1 и IN2 в соответствии с видом сравнения, выбираемым из каталога элементов.

Биты слова состояния

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

Пример

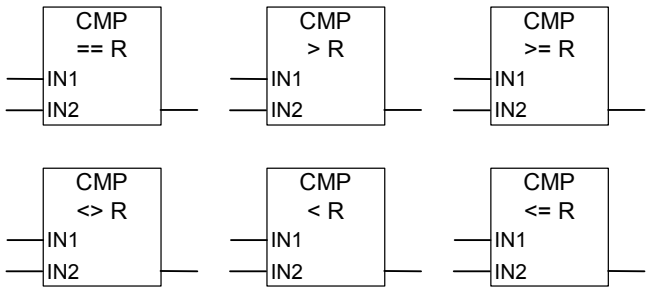


Q 4.0 устанавливается когда:

- MD0 не равен MD4
- И сигнал на входе I 0.0 равен 1.

2.4 CMP ? R : Сравнение чисел типа Real

Обозначение



Параметр	Тип данных	Область памяти	Описание
IN1	REAL	I, Q, M, D, L или константа	Первое сравниваемое значение
IN2	REAL	I, Q, M, D L или константа	Второе сравниваемое значение
Выход блока	BOOL	I, Q, M, D, L	Результат сравнения

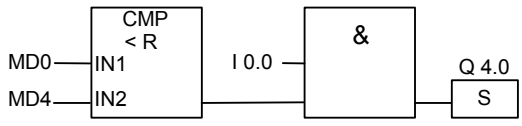
Описание

Инструкция **Сравнить числа типа Real** сравнивает значения двух 32-битных чисел с плавающей точкой. Эта инструкция сравнивает входы IN1 и IN2 в соответствии с видом сравнения, выбираемым из каталога элементов.

Биты слова состояния

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

Пример



Q 4.0 устанавливается когда:

- MD0 меньше чем MD4
- И сигнал на входе I 0.0 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

Вы можете использовать одну из следующих инструкций для формирования дополнения чисел формата integer или инверсии знака чисел с плавающей точкой:

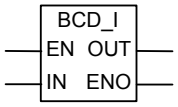
- INV_I : Инверсия числа типа Integer
- INV_DI : Инверсия числа типа Double Integer
- NEG_I : Дополнительный код числа типа Integer
- NEG_DI : Дополнительный код числа типа Double Integer
- NEG_R : Инверсия знака числа типа Real

Вы можете использовать любую из следующих инструкций для преобразования 32-битового IEEE числа с плавающей точкой в аккумуляторе 1 в 32-битовое целое (Double Integer). Отдельные инструкции отличаются методом округления:

- ROUND : Округление до двойного целого
- TRUNC : Выделение целой части
- CEIL : Округление в большую сторону
- FLOOR : Округление в меньшую сторону

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

Обозначение



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

Описание

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.

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

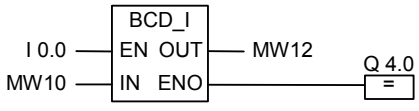
Если любая из десятичных цифр в BCD-числе находится в недопустимом диапазоне от 10 до 15, то при попытке преобразования возникает ошибка BCD, вызывающая следующую реакцию:

- CPU переходит в состояние STOP. В диагностический буфер вносится сообщение ”BCD conversion error” [“Ошибка преобразования BCD”] с идентификационным номером 2521.
- Если OB121 запрограммирован, то он вызывается.

Биты слова состояния

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

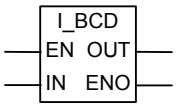
Пример



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 выполняется, то состояние сигнала выхода Q4.0 равно 1 (ENO = EN).

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

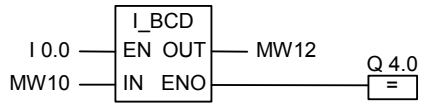
Описание

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

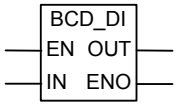
Пример



Преобразование выполняется, 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. В случае переполнения состояние сигнала на выходе Q4.0 равно 0. Если состояние сигнала на входе EN равно 0 (это значит, что преобразование не выполняется), то состояние сигнала на выходе Q4.0 тоже равно 0.

3.4 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

Инструкция преобразования **BCD 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. Выходной параметр OUT содержит результат.

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

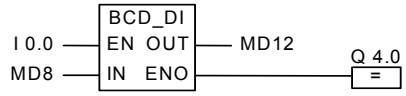
Если любая из десятичных цифр в BCD-числе находится в недопустимом диапазоне от 10 до 15, то при попытке преобразования возникает ошибка BCD, вызывающая следующую реакцию:

- CPU переходит в состояние STOP. В диагностический буфер вносится сообщение ”BCD conversion error” [“Ошибка преобразования BCD”] с идентификационным номером события 2521.
- Если OB121 запрограммирован, то он вызывается.

Биты слова состояния

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

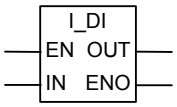
Пример



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

3.5 I _DI : Преобразование целого числа в двойное целое число

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	Преобразуемая величина
OUT	DINT	I, Q, M, D, L	Результат
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

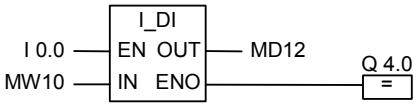
Описание

Инструкция преобразования **Целое в двойное целое** считывает содержимое входного параметра IN как целое число и преобразует его в двойное целое число. 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

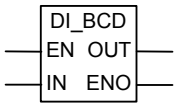
Пример



Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Содержимое меркерного слова MW10 считывается как целое число и преобразуется в двойное целое. Результат сохраняется в двойном меркерном слове MD12. Если преобразование выполняется, то состояние сигнала на выходе Q4.0 равно 1 (ENO = EN).

3.6 DI _ BCD: Преобразование двойного 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	DINT	I, Q, M, D, L или константа	Двойное целое число
OUT	DWORD	I, Q, M, D, L	BCD значение двойного целого числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

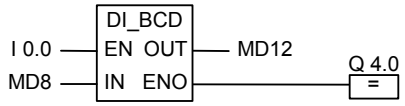
Описание

Инструкция преобразования **Двойное целое в BCD** считывает содержимое входного параметра IN как двойное целое значение и преобразует его в семизначное число в формате BCD ($\leq$ 9 999 999). Выходной параметр OUT содержит результат. В случае переполнения 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

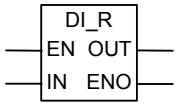
Пример



Преобразование выполняется , 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 . В случае переполнения состояние сигнала на выходе Q4.0 равно 0. Если состояние сигнала на входе EN равно 0(это значит, что преобразование не выполняется),то состояние сигнала на Q4.0 равно 0.

3.7 DI_ R: Преобразование двойного целого числа в число с плавающей точкой

Обозначение



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

Описание

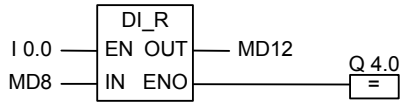
Инструкция преобразования **Двойное целое в вещественное** считывает содержимое входного параметра IN как двойное целое число и преобразует его в число с плавающей точкой. Выходной параметр OUT содержит результат. ENO всегда имеет такое же состояние сигнала, как и EN.

.

Биты слова состояния

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

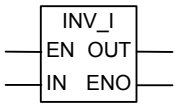
Пример



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, то состояние сигнала на выходе Q4.0 равно 0.

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 или константа	Входная величина
OUT	INT	I, Q, M, D, L	Дополнение целого числа до единицы
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

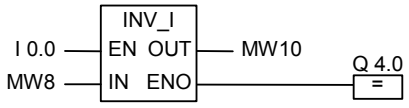
Описание

Инструкция **Инверсия целого числа** считывает содержимое входного параметра IN и выполняет поразрядную логическую операцию Исключающее ИЛИ (см. раздел 15.6) 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



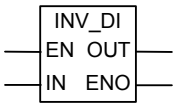
Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Значение каждого бита в MW8 инвертируется:

MW8 = 01000001 10000001 → MW10 = 10111110 01111110

Преобразование не выполняется, когда состояние сигнала на I0.0 равно 0. (ENO = EN).

3.9 INV_DI: Инверсия двойного целого числа

Обозначение



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

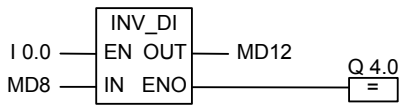
Описание

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



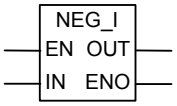
Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Значение каждого бита двойного меркерного слова MD8 инвертируется:

MD8 = F0FF FFF0 $\rightarrow$ MD12 = 0F00 000F

Если преобразование не выполняется , то выход IQ4.0 равно 0 (ENO = EN).

3.10 NEG_I: Дополнительный двоичный код 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 или константа	Входная величина
OUT	INT	I, Q, M, D, L	Дополнение целого числа до двух
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

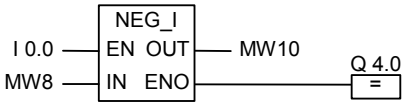
Описание

Инструкция **Дополнительный двоичный код целого числа** считывает содержимое входного параметра IN и изменяет знак (например, с положительного значения на отрицательное). 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



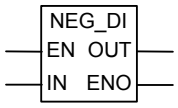
Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Значение меркерного слова MW8 выводится на OUT в меркерное словоMW10 с противоположным знаком. Пример:

MW8 = +10 $\rightarrow$ MW10 = -10

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

3.11 NEG_DI: Дополнительный двоичный код двойного целого числа

Обозначение



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

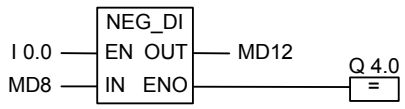
Описание

Инструкция **Дополнительный двоичный код двойного целого числа** считывает содержимое входного параметра IN и изменяет знак (например, с положительного значения на отрицательное). 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.

Битв слова состояния

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

Пример



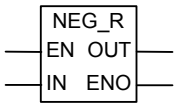
Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Значение двойного меркерного слова MD8 выводится на OUT в двойное меркерное слово MD10 с противоположным знаком

MW8 = +10 $\rightarrow$ MW10 = -10

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

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 или константа	Входная величина
OUT	REAL	I, Q, M, D, L	Результат есть входная величина с противоположным знаком
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

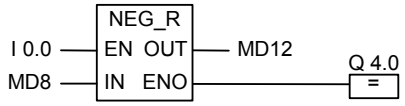
Описание

Инструкция **Изменить знак числа типа REAL** считывает содержимое входного параметра IN и инвертирует знаковый бит (инструкция меняет знак числа, например. с 0 для плюса на 1 для минуса). Биты экспоненты и мантиссы остаются неизменными. Выходной параметр OUT выдает результат. ENO всегда имеет то же состояние сигнала, что и EN, кроме случая, когда состояние сигнала на EN равно 1 и происходит переполнение. В этом случае состояние сигнала на ENO равно 0.

Биты слова состояния

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

Пример



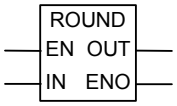
Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Значение двойного меркерного слова MD8выводится на OUT в двойное меркерное слово MD12 с противоположным знаком, как показано в примере:

MD8 = + 6.234 $\rightarrow$ MD12 = - 6.234

Если преобразование не выполняется, то состояние сигнала на выходе Q4.0 равно 0 (ENO = EN).

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

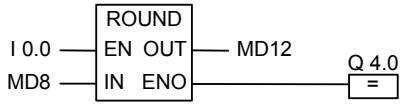
Описание

Инструкция **Округлить до двойного целого** считывает содержимое входного параметра IN как вещественное число и преобразует его в двойное целое 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 устанавливается в 0. Если входная величина не является вещественным числом, то биты OV и OS имеют значение 1, а ENO имеет значение 0.

Биты слова состояния

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

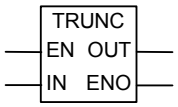
Пример



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. Если происходит переполнение, то состояние сигнала на выходе Q4.0 равно 0. Если состояние сигнала на входе EN равно 0 (это значит, что преобразование не выполняется), то состояние сигнала на выходе Q4.0 тоже равно 0.

3.14 TRUNC : Выделение целой части из 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 или константа	Округляемая величина
OUT	DINT	I, Q, M, D, L	Целая часть IN
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

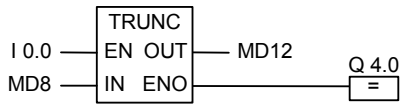
Описание

Инструкция **Выделение целой части из числа типа REAL** считывает содержимое входного параметра IN как вещественное число и преобразует это число в двойное целое число (например, 1,5 становится 1). Результат является целой частью вещественного числа. Он содержится в выходном параметре OUT. Если происходит переполнение, то ENO устанавливается в 0. Если входная величина не является вещественным числом, то биты OV и OS имеют значение 1, а ENO имеет значение 0.

Биты слова состояния

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

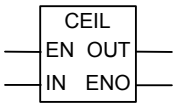
Пример



Преобразование выполняется, если состояние сигнала на I0.0 равно 1. Содержимое двойного меркерного слова MD8 считывается как вещественное число и преобразуется в двойное целое число в соответствии с принципом «округления до нуля». Результатом является целая часть числа, которая сохраняется в двойном меркерном слове MD12. Если происходит переполнение, то состояние сигнала на выходе Q4.0 равно 0. Если состояние сигнала входе EN равно 0 (это значит, что преобразование не выполняется), то состояние на выходе Q4.0 тоже равно 0.

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 или константа	Преобразуемая величина
OUT	DINT	I, Q, M, D, L	Результат
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

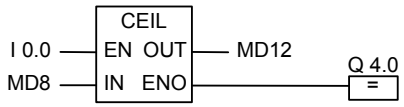
Описание

Инструкция **Округление до ближайшего большего целого числа** считывает содержимое входного параметра IN как вещественное число и преобразует это число в двойное целое число (например, +1,2 → +2; -1, 5 → -1). Результатом является наименьшее целое число, большее или равное заданному вещественному числу. Выходной параметр OUT содержит результат. Если происходит переполнение, то ENO равно 0. Если входная величина не является вещественным числом, то биты OV и OS имеют значение 1, а ENO имеет значение 0.

Биты слова состояния

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

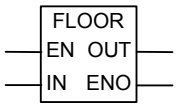
Пример



Преобразование выполняется, если I0.0 равен 1. Содержимое двойного меркерного слова MD8 считывается как вещественное число и преобразуется в двойное целое число округлением до ближайшего большего (или равного) целого числа. Результат со храняется в двойном меркерном слове MD12. Если происходит переполнение, то состояние сигнала на выходе Q4.0 равно 0. Если состояние сигнала на выходе Q4.0 равно 0. Если состояние сигнала на входе EN равно 0 (преобразование не выполняется) то состояние сигнала на выходе Q4.0 тоже равно 0.

3.16 FLOOR: Округление до ближайшего меньшего 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 или константа	Преобразуемая величина
OUT	DINT	I, Q, M, D, L	Результат
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

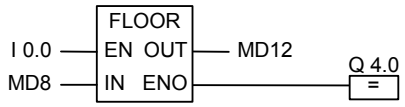
Описание

Инструкция **Округление до ближайшего меньшего целого числа** считывает содержимое входного параметра IN как вещественное число и преобразует это число в двойное целое число Результатом является наименьшее целое число, меньшее или равное заданному вещественному числу. Выходной параметр OUT содержит результат. Если происходит переполнение, то ENO устанавливается в 0. Если входная величина не является вещественным числом, то биты OV и OS имеют значение 1, а ENO имеет значение 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. Результат сохраняется в меркерном двойном слове MD12. При переполнении, выход Q4.0 устанавливается в 0. При подаче на вход EN сигнала 0 (означает что преобразование не выполняется), значение выхода Q4.0 будет также в 0.

4 Инструкции счетчиков

4.1 Обзор инструкций счетчиков

Область памяти

Счетчики имеют область, зарезервированную для них в памяти Вашего CPU. Эта область памяти резервирует по одному 16-битному слову для каждого адреса счетчика. При программировании в FBD 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, когда счетчик устанавливается. Значение счетчика может находиться в диапазоне от 0 до 999.

Вы можете изменять значение счетчика, используя следующие инструкции:

- S_CUD : Назначение параметров и прямой/обратный счет
- S_CU : Назначение параметров и прямой счет
- S_CD : Назначение параметров и обратный счет
- SC : Назначение параметров
- CU : Прямой счет
- CD : Обратный счет

Конфигурация 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

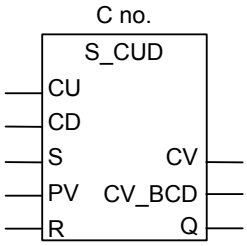
C# означает двоично-десятичный формат (BCD-формат: каждая группа из четырех битов содержит двоичный код для одного десятичного разряда).

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

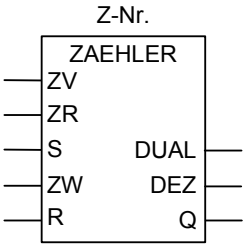
4.2 S_CUD: Назначение параметров и прямой/обратный счет

Обозначение

Английский



Немецкий



Параметры English	**Параметры German**	**Тип данных**	**Область памяти**	**Описание**
no.	Nr.	COUNTER	C	Номер счетчика .Диапазон номеров зависит от CPU.
CU	ZV	BOOL	I, Q, M, D, L	ZV вход : прямой счет
CD	ZR	BOOL	I, Q, M, D, L	ZR вход: обратный счет
S	S	BOOL	I, Q, M, D, L, T, C	Вход предустановки счетчика
PV	ZW	WORD	I, Q, M, D, L или константа	Значение счетчика от 0 до 999 или Значение счетчика, введенное 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

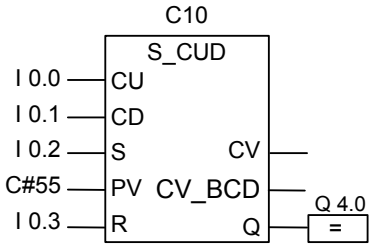
Нарастающий фронт (изменение сигнала с 0 на 1) на входе S устанавливает счетчик **прямого/обратного счета** на значение, указанное на входе предварительного задания PV. Счетчик увеличивается на 1, если состояние сигнала на входе CU изменяется с 0 на 1 (нарастающий фронт) и значение счетчика меньше 999. Счетчик уменьшается на 1, если состояние сигнала на входе CD изменяется с 0 на 1 (нарастающий фронт) и значение счетчика больше 0. Если имеет место нарастающий фронт на обоих счетных входах, то выполняются обе инструкции и счетчик сохраняет прежнее значение. Счетчик сбрасывается, если нарастающий фронт появляется на входе R.
Сброс счетчика устанавливает его значение в 0.

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

Пример



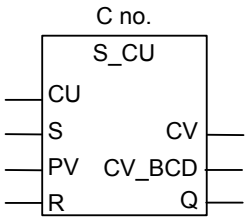
Изменение состояния сигнала с 0 на 1 на входе I0.2 устанавливает C10 со значением 55. Если состояние сигнала на входе I0.0 меняется с 0 на 1, то значение счетчика C10 увеличивается на 1, кроме случая, когда значение счетчика C10 уже равно 999.Если вход I0.1 меняется с 0нао 1,то счетчик C10 уменьшается на 1кроме 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 C 10 не равен 0.

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

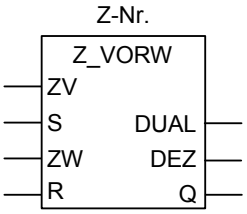
4.3 S_CU: Назначение параметров и прямой счет

Обозначение

Английский



Немецкий



Параметр English	Параметр German	Тип данных	Область памяти	Описание
no.	Nr.	COUNTER	C	Номер счетчика. Диапазон номеров зависит от CPU.
CU	ZV	BOOL	I, Q, M, D, L	Вход ZV : прямой счет
S	S	BOOL	I, Q, M, D, L, T, C	Вход для предустановки счетчика
PV	ZW	WORD	I, Q, M, D, L or constant	Значение счетчика в диапазоне от 0 до 999 или Значение счетчика введенное как C#<value> в форматеBCD
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

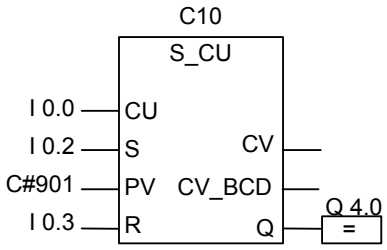
Нарастающий фронт (изменение сигнала с 0 на 1) на входе S инструкции **Прямой счет** устанавливает счетчик значением, указанным на входе предварительного задания PV. При нарастающем фронте на входе CU значение счетчика увеличивается на 1, если значение счетчика меньше 999. Счетчик сбрасывается нарастающим фронтом на входе R. Сброс счетчика устанавливает его значение в 0.

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

Пример



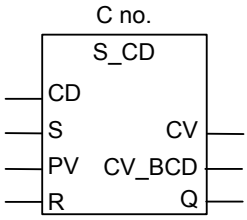
Изменение сигнала с 0 на 1 на входе I0.2 устанавливает счетчик C 10 на значение 901.Если состояние сигнала I0.0 изменяется с 0 на 1, значение счетчика C10 увеличивается на 1 до тех пор, пока не станет равным 999.Если вход I0.3 принимает значение 1, то содержимое счетчика C10 сбрасывается в 0. Состояние сигнала на выходе Q4.0 равно 1, если значение счетчика C10 отлично от 0.

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

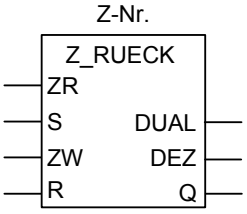
4.4 S_CD: Обратный счет

Обозначение

Английский



Немецкий



Параметр English	**Параметр German**	**Тип данных**	**Область памяти**	**Описание**
no.	Nr.	COUNTER	C	Номер счетчика. Диапазон номеров зависит от CPU.
CD	ZR	BOOL	I, Q, M, D, L	Вход ZR : обратный счет
S	S	BOOL	I, Q, M, D, L, T, C	Вход для предустановки счетчика
PV	ZW	WORD	I, Q, M, D, L or constant	Значение счетчика в диапазоне от 0 до 999 или Значение счетчика введенное как C#<value> в форматеBCD
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

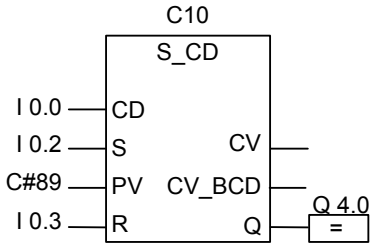
Нарастающий фронт (изменение сигнала с 0 на 1) на входе S инструкции **Обратный счет** устанавливает счетчик значением, указанным на входе предварительного задания PV. При нарастающем фронте на входе CD значение счетчика уменьшается на 1, если значение счетчика больше 0. Счетчик сбрасывается нарастающим фронтом 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

Пример



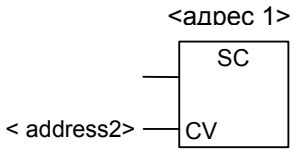
Изменение сигнала с 0 на 1 на входе I0.2 устанавливает счетчик C 10 на значение 89. Если состояние сигнала I0.0 изменяется с 0 на 1, значение счетчика C10 уменьшается на 1 до тех пор, пока не станет равным 0. Если вход I0.3 принимает значение 1, то содержимое счетчика C10 сбрасывается в 0. Состояние сигнала на выходе Q4.0 равно 1, если значение счетчика C10 отлично от 0.

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

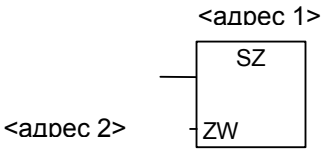
4.5 SC : Установка значения счетчика

Обозначение

Английский



Немецкий



Параметр English	Параметр German	Тип данных	Область памяти	Описание
Счетчик №	Счетчик №	COUNTER	C	Адрес1 задает номер счетчика, который получает заданное значение.
CV	ZW	WORD	I, Q. M, D, L или константа	Предустановленное значение (адрес 2) может быть в пределах между 0 и 999. Когда Вы вводите константу, идентификаторC# должен ставиться перед задаваемым значением , например, C#100.

Описание

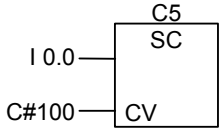
Инструкция **Установка значения счетчика** назначает счетчику предустановленное значение при появлении нарастающего фронта RLO.

Вы можете разместить блок **Установка значения счетчика** только в правой крайней позиции логической цепочки с указанием номера задаваемого счетчика.

Биты слова состояния

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

Пример

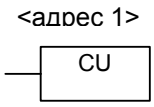


Счетчик C5 получает значение 100 если сигнал на входе I0.0 изменяется с 0 на 1 (нарастающий фронт RLO). Формат C# показывает, что Вы вводите значение в BCD коде. Без нарастающего фронта RLO содержимое счетчика C5 не меняется.

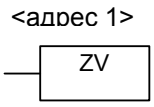
4.6 CU : Счет на увеличение

Обозначение

Английский



Немецкий



Параметр	Тип данных	Область памяти	Описание
Счетчик №.	COUNTER	C	Адрес определяет номер счетчика для увеличения его содержимого.

Описание

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.

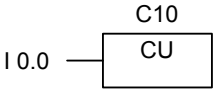
Инструкция **Установка счетчика** задает значение счетчика.

Вы можете разместить блок **Счет на увеличение** только в правой крайней позиции логической цепочки с указанием номера задаваемого счетчика.

Биты слова состояния

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

Пример

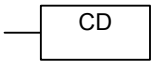


Счетчик C10 увеличивает свое содержимое на 1 , если сигнал на входе I0.0 изменяется с 0 на 1 (пока оно не достигло 999). Без нарастающего фронта RLO содержимое счетчика C10 не меняется.

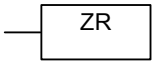
4.7 CD : Счет на уменьшение

Обозначение

Английский



Немецкий



Параметр	Тип данных	Область памяти	Описание
Счетчик №.	COUNTER	C	Адрес определяет номер счетчика для уменьшения его содержимого.

Описание

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

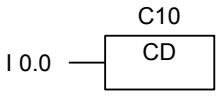
Инструкция **Установка счетчика** задает значение счетчика.

Вы можете разместить блок **Счет на уменьшение** только в правой крайней позиции логической цепочки с указанием номера задаваемого счетчика.

Биты слова состояния

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

Пример



Счетчик C10 уменьшает свое содержимое на 1 , если сигнал на входе I0.0 изменяется с 0 на 1 (пока оно не достигло 0). Без нарастающего фронта RLO содержимое счетчика C10 не меняется.

5 Инструкции с блоками данных

5.1 OPN : Открыть блок данных

Обозначение

<DB-номер>или
<DI-номер>

Параметр	Тип данных	Область памяти	Описание
Номер DB или DI	BLOCK_DB	-	Номер блока данных в регистре DB или DI; Допустимые значения зависят от типа CPU.

Описание

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

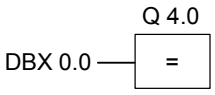
Биты слова состояния

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

Пример

Network 1

Network 2



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 Условный переход по нулю

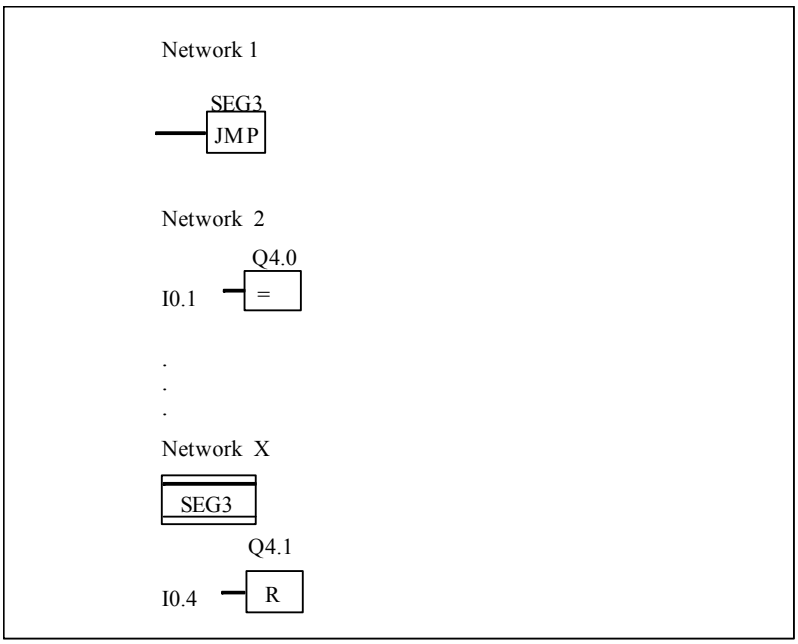
Метка перехода как операнд

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 должна находиться в начале сегмента. Целевая метка вводится в начале сегмента выбором LABEL из каталога элементов FBD. Появляется пустой блок. В этом блоке записывается имя метки



6.2 JMP: Безусловный переход в блоке

Обозначение

Операнд

Параметр	Тип данных	Область памяти	Описание
Имя метки перехода	-	-	Адрес указывает метку, на которую программа должна перейти безусловно.

Описание

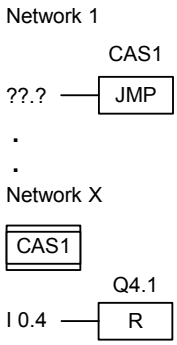
Инструкция **Безусловный переход в блоке** соответствует команде “перейти на метку”. Ни одна из команд, расположенных между командой перехода и меткой, не выполняется.

Эту инструкцию можно использовать во всех логических блоках, например, в организационных блоках (OB), в функциональных блоках (FB) и в функциях (FC).

Биты слова состояния (инструкция не изменяет эти биты)

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

Пример



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

6.3 JMP: Условный переход в блоке

Обозначения

<Адрес>

Параметр	Тип данных	Область памяти	Описание
Имя метки перехода	-	-	Адрес указывает метку, на которую программа должна перейти при RLO равном 1 .

Описание

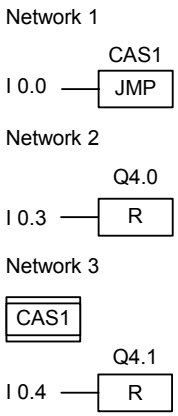
Инструкция **Условный переход в блоке** соответствует команде “перейти на метку”, если RLO равен 1. Для этой инструкции тоже используется элемент FBD “Безусловный переход”, но он делается условным предшествующей логической операцией. Условный переход выполняется только тогда, когда результат логической инструкции равен 1. Ни одна из команд между командой перехода и меткой не выполняется.

Эту инструкцию можно использовать во всех логических блоках, например, в организационных блоках (OB), в функциональных блоках (FB) и в функциях (FC).

Биты слова состояния

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

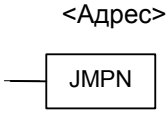
Пример



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 I0.3равно1.

6.4 JMPN: Переход при 0

Обозначение



Параметр	Тип данных	Область памяти	Описание
Имя метки перехода	-	-	Адрес указывает метку, на которую программа должна перейти при RLO равном 0 .

Описание

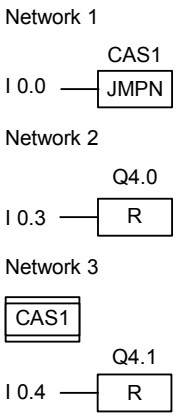
Инструкция **Переход при 0** соответствует команде “перейти на метку”, которая выполняется, если RLO равен 0.

Эту инструкцию можно использовать во всех логических блоках, например, в организационных блоках (OB), в функциональных блоках (FB) и в функциях (FC).

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 равно1

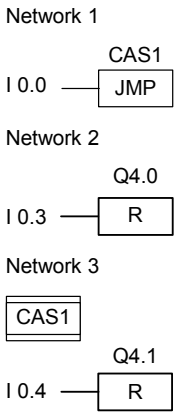
6.5 LABEL: Метка перехода

Обозначение

Описание

Метка перехода - это идентификатор места назначения инструкции перехода. 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

см.также оценку битов слова состояния в математических инструкциях с целыми числами

7.2 Оценка битов слова состояния в случае арифметических операций с целыми числами

Описание

Математические инструкции с целыми числами приводят к изменениям следующих бит слова состояния: CC 1 и CC 0, OV иOS.

Следующие таблицы показывают изменения бит 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	*

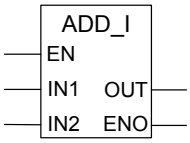
* Бит OS не изменяется при выполнении этих инструкций

Недопустимый диапазон значения	**A1**	**A0**	**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	**A1**	**A0**	**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, D, L, T, C	Деблокировка входа
IN1	INT	I, Q, M, D, L или константа	Первое значение для математической инструкции
IN2	INT	I, Q, M, D, L или константа	Второе значение для математической инструкции
OUT	INT	I, Q, M, D, L	Результат выполнения математической инструкции
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

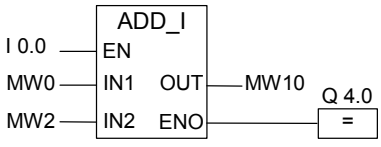
Описание

Состояние сигнала 1 на входе EN (деблокировка входа) активизирует инструкцию **Сложить целые числа**. Эта инструкция складывает 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.

Биты слова состояния

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

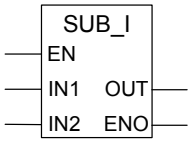
Пример



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 принимает значение 0 инструкция не выполняется .

7.4 SUB_I : Вычитание целых чисел

Обозначение



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

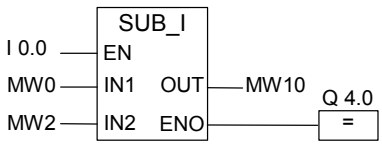
Описание

Состояние сигнала 1 на входе EN (деблокировка входа) активизирует инструкцию **Вычитание 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.

Биты слова состояния

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

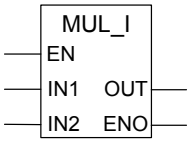
Пример



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 принимает значение 0 и инструкция не выполняется .

7.5 MUL_I : Умножение целых чисел

Обозначение



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

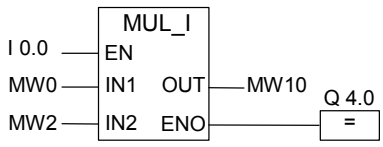
Описание

Состояние сигнала 1 на входе EN (деблокировка входа) активизирует инструкцию **Умножение целых чисел.** Эта инструкция умножает 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.

Биты слова состояния

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

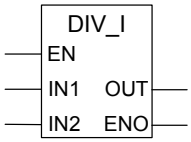
Пример



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 или состояние сигнала на входе I0.0 равно 0, выход Q4.0 принимает значение 0 и инструкция не выполняется .

7.6 DIV_I : Деление целых чисел

Обозначение



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

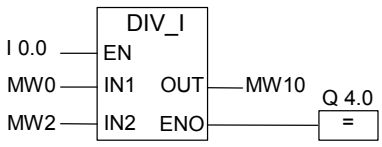
Описание

Состояние сигнала 1 на входе EN (деблокировка входа) активизирует инструкцию **Деление целых чисел.** Эта инструкция делит значения 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.

Биты слова состояния

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

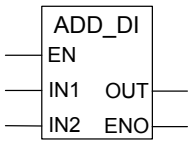
Пример



Сигнал 1 на входе I0.0 активирует блок DIV_I . Результат деления MW0 x 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 принимает значение 0 и инструкция не выполняется

7.7 ADD_DI: Сложение двойных целых чисел

Обозначение



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

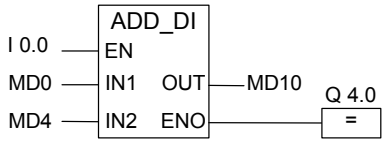
Описание

Состояние сигнала 1 на входе EN (деблокировка входа) активизирует 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

Биты слова состояния

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

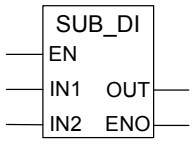
Пример



Сигнал 1 на входе I0.0 активирует блок ADD_DI . Результат сложения MD0 + MD2 передается в меркерное слово MD 10. Если результат выходит за пределы допустимого диапазона для двойных целых чисел или состояние сигнала на входе I0.0 равно 0, выход Q4.0 принимает значение 0 и инструкция не выполняется.

7.8 SUB_DI : Вычитание двойных целых чисел

Обозначение



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

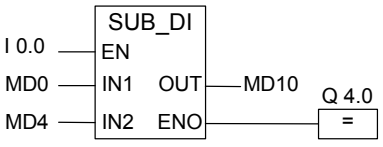
Описание

Состояние сигнала 1 на входе EN (деблокировка входа) активизирует 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.

Биты слова состояния

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

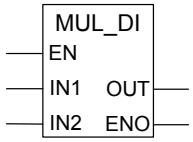
Пример



Сигнал 1 на входе I0.0 активирует блок SUB_DI. Результат вычитания 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 принимает значение 0 и инструкция не выполняется .

7.9 MUL_DI: Умножение двойных целых чисел

Обозначение



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

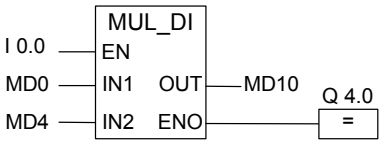
Описание

Состояние сигнала 1 на входе EN (деблокировка входа) активизирует инструкцию **Умножение двойных целых чисел**. 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.

Биты слова состояния

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

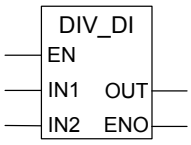
Пример



Сигнал 1 на входе I0.0 активирует блок MUL_DI . Результат умножения MD0 x MD2 передается в меркерное слово MD10. Если результат выходит за пределы допустимого диапазона для двойных целых целых чисел или состояние сигнала на входе I0.0 равно 0, выход Q4.0 принимает значение 0 и инструкция не выполняется .

7.10 DIV_DI: Деление двойных целых чисел

Обозначение



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

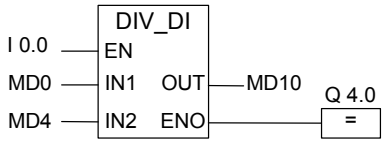
Описание

Состояние сигнала 1 на входе EN (деблокировка входа) активизирует 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.

Биты слова состояния

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

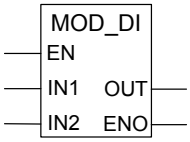
Пример



Сигнал 1 на входе I0.0 активирует блок DIV_DI . Результат деления MD0 / MD2 передается в меркерное слово MD10. Если результат выходит за пределы допустимого диапазона для двойных целых целых чисел или состояние сигнала на входе I0.0 равно 0, выход Q4.0 принимает значение 0 и инструкция не выполняется.

7.11 MOD_DI: Получение остатка от деления двойных целых чисел

Обозначение



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

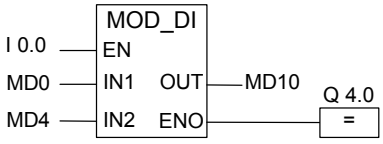
Описание

Состояние сигнала 1 на входе EN (деблокировка входа) активизирует 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.

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 / MD2 передается в меркерное слово MD10. Если результат выходит за пределы допустимого диапазона для двойных целых целых чисел или состояние сигнала на входе I0.0 равно 0, выход Q4.0 принимает значение 0 и инструкция не выполняется.

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

8.2 Анализ битов слова состояния в операциях с плавающей точкой

Описание

Инструкции с плавающей точкой влияют на следующие биты слова состояния:

CC1 и CC , OV и OS .

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

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	*

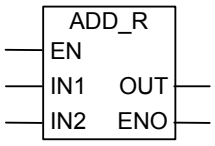
* Бит OS не изменяется этими инструкциями.

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, D, L, T, C	Деблокировка входа
IN1	REAL	I, Q, M, D, L или константа	Первое слагаемое
IN2	REAL	I, Q, M, D, L или константа	Второе слагаемое
OUT	REAL	I, Q, M, D, L	Результат сложения
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

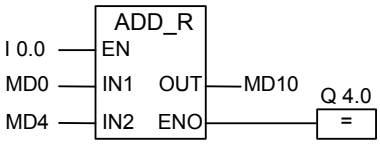
Описание

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.

Биты слова состояния

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

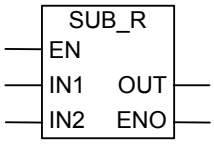
Пример



Состояние сигнала 1 на входе I0.0 активирует блок ADD_R. Результат сложения MD0 + MD4 выводится в двойное меркерное слово MD10. Если 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 0 и команда не выполняется.

8.3.2 SUB_R: Вычитание 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	Уменьшаемое
IN2	REAL	I, Q, M, D, L или константа	Вычитаемое
OUT	REAL	I, Q, M, D, L	Результат вычитания
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

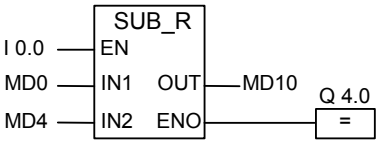
Описание

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.

Биты слова состояния

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

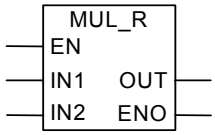
Пример



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 0 и команда не выполняется.

8.3.3 MUL_R: Умножение 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	Первый множитель
IN2	REAL	I, Q, M, D, L или константа	Второй множитель
OUT	REAL	I, Q, M, D, L	Произведение
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

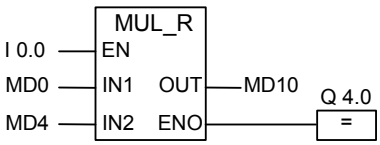
Описание

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.

Биты слова состояния

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

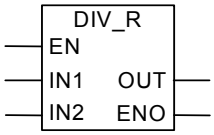
Пример



Если состояние сигнала 1 на входе I0.0 активизирует блок MUL_R . Результат умножения MD0 x MD4 выводится 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 0 и команда не выполняется.

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

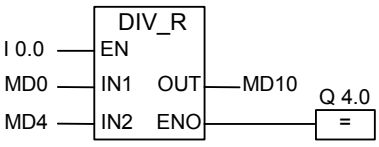
Описание

Состояние сигнала 1 на входе EN (деблокировка входа) активизирует 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

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

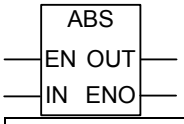


Состояние сигнала 1 на входе I0.0 активирует блок DIV_R. Результат деления MD0 на MD4 выводится в двойное меркерное слово MD10. Если один из входов или результат не является числом с плавающей точкой или если состояние сигнала на входе I0.0 равно 0, то выход Q4.0 устанавливается в 0 и команда не выполняется.

.

8.3.5 ABS: Образование абсолютного значения числа с плавающей точкой

Обозначение



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

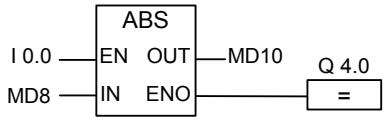
Описание

С помощью инструкции **Образование абсолютного значение числа с плавающей точкой** Вы можете найти абсолютную величину числа с плавающей точкой.

Биты слова состояния

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

Пример



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

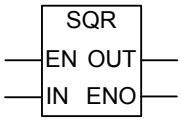
MD8 = +6.234 дает в результате MD12 = 6.234 x 1.

Выход Q4.0 равно 0 , 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, D, L, T, C	Деблокировка входа
IN	REAL	I, Q, M, D, L или константа	Число
OUT	REAL	I, Q, M, D, L	Квадрат числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

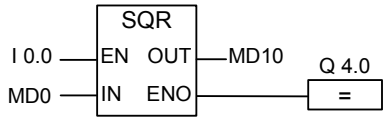
Описание

С помощью инструкции **Вычисление квадрата числа с плавающей точкой,** Вы можете возвести число с плавающей точкой в квадрат. Если вход IN или результат не является числом с плавающей точкой, то биты OV и OS устанавливаются в 1, а ENO устанавливается в 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

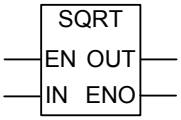
Пример



Состояние сигнала 1 на входе I0.0 активирует блок SQR. Результат возведения MD0 в квадрат выводится в двойное меркерное слово MD10. Если MD0 меньше 0 или если вход или результат не является числом с плавающей точкой или если состояние сигнала на входе I0.0 равно 0, то выход Q4.0 равен 0.

8.4.2 SQRT : Вычисление квадратного корня числа с плавающей точкой

Обозначение



Параметр	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	Число
OUT	REAL	I, Q, M, D, L	Квадратный корень из числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

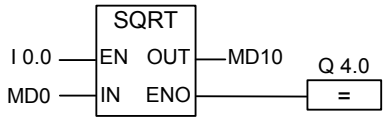
Описание

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 операнда больше 0. Если вход или результат не является числом с плавающей точкой, то биты OV и OS устанавливаются в 1, а ENO устанавливается в 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

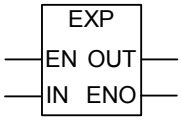
Пример



Состояние сигнала 1 на входе I0.0 активирует блок SQRT . Результат извлечения квадратного корня из MD0 выводится в двойное меркерное слово MD10. Если MD0 меньше 0 или если вход или результат не является числом с плавающе точкой или если состояние сигнала на входе I0.0равно 0, то выход Q4.0 равен 0.

8.4.3 EXP: Вычисление экспоненты числа с плавающей точкой

Обозначение



Параметр	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	Число
OUT	REAL	I, Q, M, D, L	Экспоненциальное значение числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

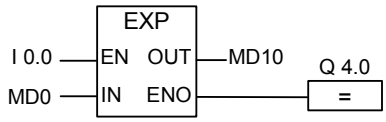
Описание

С помощью инструкции **Вычисление экспоненты числа с плавающей точкой** , Вы можете найти экспоненциальное значение числа с плавающей точкой по основанию e (=2,71828...). Если вход или результат не является числом с плавающей точкой, то биты OV и OS устанавливаются в 1, а ENO устанавливается в 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

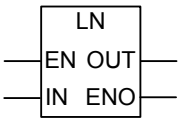
Пример



Состояние сигнала 1 на входе I0.0 активирует блок EXP . Результат получения эспоненциального значения MD0 выводится в двойное меркерное слово MD10. Если MDO меньше 0 или если вход или результат не является числом с плавающей точкой или если состояние сигнала на входе I 0.0 равно 0, то выход Q4.0 равне 0.

8.4.4 LN : Вычисление натурального логарифма числа с плавающей точкой

Обозначение



Параметр	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	Число
OUT	REAL	I, Q, M, D, L	Натуральный логарифм числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

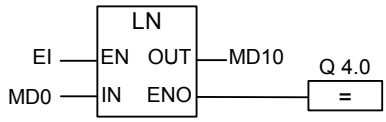
Описание

С помощью инструкции **Вычисление натурального логарифма числа с плавающей точкой** ,Вы можете найти натуральный логарифм числа с плавающей точкой. Если вход или результат не является числом с плавающей точкой, то биты OV и OS устанавливаются в 1, а ENO устанавливается в 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



Состояние сигнала 1 на входе I0.0 активирует блок LN . Результат логарифмирования MD0 выводится в меркерное слово MD10. Если MD0 меньше 0, или если вход или результат не является числом с плавающей точкой, или если состояние сигнала на входе I0.0 равно 0, то выход Q4.0 равен 0.

8.4.5 Вычисление тригонометрических функций углов в виде чисел с плавающей точкой

Описание

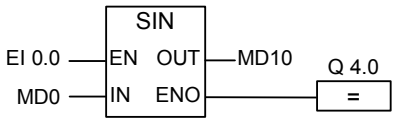
С помощью следующих инструкций Вы можете вычислить тригонометрические функции углов, представленные в виде 32-битных чисел с плавающей точкой в формате IEEE.

Инструкция	Значение
SIN	Вычисляет синус числа с плавающей точкой, представляющего угол, выраженный в радианах.
COS	Вычисляет косинус числа с плавающей точкой, представляющего угол, выраженный в радианах.
TAN	Вычисляет тангенс числа с плавающей точкой, представляющего угол, выраженный в радианах.
ASIN	Вычисляет арксинус числа с плавающей точкой. Результат есть угол, выраженный в радианах. Значение находится в следующем диапазоне: $-\pi / 2 \leq$ арксинус $\leq + \pi / 2$, где $\pi = 3{,}14...$
ACOS	Вычисляет арккосинус числа с плавающей точкой. Результат есть угол, выраженный в радианах. Значение находится в следующем диапазоне: $0 \leq$ арккосинус $\leq + \pi$, где $\pi = 3.14...$
ATAN	Вычисляет арктангенс числа с плавающей точкой. Результат есть угол, выраженный в радианах. Значение находится в следующем диапазоне: $-\pi / 2 \leq$ арктангенс $\leq + \pi / 2$, где $\pi = 3.14...$

Биты слова состояния

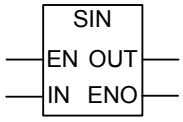
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	X	X	X	X	X	0	X	X	1

Обозначение



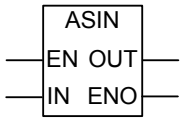
Состояние сигнала 1 на входе I0.0 активирует блок SIN . Синус MD0 выводится в двойное меркерное слово MD10. Если MD0 меньше 0 или если вход или результат не является числом с плавающей точкой или если состояние сигнала на входе I0.0 равно 0, то выход Q4.0 равен 0.

Обозначение



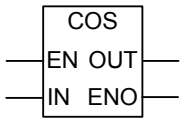
Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	REAL	I, Q, M, D, L или константа	Число
OUT	REAL	I, Q, M, D, L	Синус числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

Обозначение



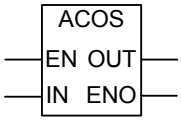
Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	REAL	I, Q, M, D, или константа	Число
OUT	REAL	I, Q, M, D, L	Арксинус числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

Обозначение



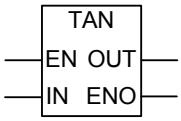
Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	REAL	I, Q, M, D, L или константа	Число
OUT	REAL	I, Q, M, D, L	Косинус числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

Обозначение



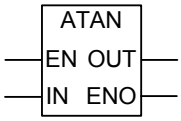
Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	REAL	I, Q, M, D, L или константа	Число
OUT	REAL	I, Q, M, D, L	Арккосинус числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

Обозначение



Параметр	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	Число
OUT	REAL	I, Q, M, D, L	Тангенс числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

Обозначение

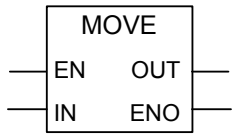


Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, D, L, T, C	Деблокировка входа
IN	REAL	I, Q, M, D, L или константа	Число
OUT	REAL	I, Q, M, D, L	Арктангенс числа
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

9 Инструкции передачи

9.1 MOVE : Передача значения

Обозначение



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

Описание

С помощью команды **Передать значение,** Вы можете присваивать конкретные значения переменным.

Значение, указанное на входе IN, копируется в адрес, указанный на выходе OUT. ENO имеет то же состояние сигнала, что и EN.

С помощью блока MOVE команда **Передать значение** может копировать все типы данных длиной 8, 16 или 32 бита. Типы данных, определенные пользователем, такие как массивы или структуры, должны копироваться с помощью системной функции SFC 20 ”BLKMOV” .На команду **Передать значение** оказывает влияние Главное управляющее реле (MCR). За дополнительной информацией о функциях MCR обращайтесь к разделу Вкл./Выкл. Master Control Relay.

Биты слова состояния

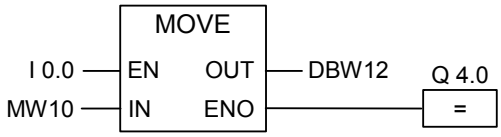
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	1	-	-	-	-	0	1	1	1

Замечание

При передаче значений между переменными различных типов данных, старшие байты отсекаются или заполняются нулями:

Пример: Двойное слово	**1111 1111**	**0000 1111**	**1111 0000**	**0101 0101**
Передать	**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**
Передать	**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.

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 Number >

Параметр	Тип данных	Область памяти	Описание
Number	BLOCK_FC	-	Номер FC или SFC (напр., FC10 или SFC59). Доступные SFCs зависят от Вашего CPU. Условный вызов с параметром типа BLOCK_FC в качестве адреса возможен в FB, но не в FC.

Описание

С помощью инструкции **Вызвать FC/SFC без параметров** можно вызвать функцию (FC) или системную функцию (SFC), которые не имеют параметров. Вызов является условным или безусловным в зависимости от предшествующей логической инструкции (см. пример).

В кодовом разделе функции (FC) Вы не можете задавать никаких параметров типа BLOCK_FC в качестве адреса для условного вызова. Однако, Вы можете задать параметр типа BLOCK_FC в качестве адреса в функциональном блоке (FB).

Условный вызов выполняется только в том случае, если RLO равен 1. Если условный вызов не выполняется, то RLO после инструкции вызова равен 1. Если инструкция выполняется, то реализуются следующие функции:

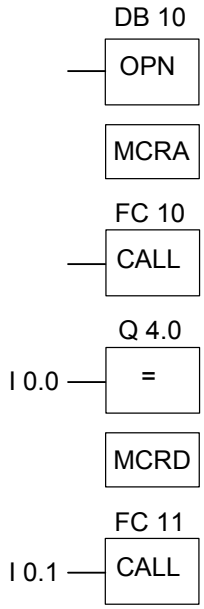
- Сохраняется адрес, необходимый для возврата в вызывающий блок.
- Сохраняются регистры блоков данных (блока данных и экземпляра блока данных).
- Предыдущая область локальных данных заменяется текущей областью локальных данных.
- Бит MA (бит активизации MCR) записывается в стек блоков (BSTACK).
- Для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

Затем исполнение программы продолжается в вызванном блоке.

Биты слова состояния

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

Пример



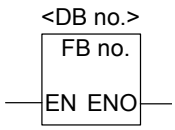
Если выполняется безусловный вызов для FC10, то команда CALL реализует следующие функции:

- Сохраняет адрес, необходимый для возврата в текущий FB.
- Сохраняет регистры для DB10 и для экземпляра блока данных FB.
- Помещает в стек блоков (BSTACK) бит MA, установленный в 1 в команде MCRA , и сбрасывет этот бит в 0 для вызванного FC10

Исполнение программы продолжается в FC10. Если Вы хотите использовать функцию MCR в FC10, Вы должны ее там повторно ативизировать. Когда FC10 завершается, исполнение программы возвращается в вызывающий 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.

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),
- Создание новой области в локальном стэке для вызываемого функционального блока.
- Бит MA (бит активности функции MCR) сохраняется в BSTACK,

Биты слова состояния

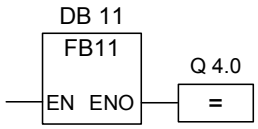
		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Условный вызов	Записывает	X	-	-	-	0	0	X	X	X
Безусловный вызов	Записывает	-	-	-	-	0	0	X	X	X

Пример

Network 1

Network 2

Network 3



Network 4

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:

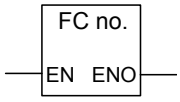
Сохраняются точка возврата в вызывающий блок и регистры блоков данных для DB10 и экземплярного блока вызывающего FB. Бит MA , установленный в 1 инструкцией MCRA сохраняется в стэке блоков BSTACK и сбрасывается в 0 для вызванного блока FB11. Выполнение программы продолжается в FB11. Если FB11 требуется активация MCR, то это должно быть повторно сделано в функциональном блоке. Значение RLO должно быть сохранено в бите BR с помощью инструкции [SAVE] для возможности 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 стэке для вызываемой функции.
- Бит MA (бит активности функции MCR) сохраняется в BSTACK,

Наконец, 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 назначены 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

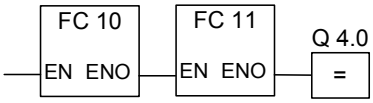
		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Условный вызов	Записывает	X	-	-	-	0	0	X	X	X
Безусловный вызов	Записывает	-	-	-	-	0	0	X	X	X

Пример

Network 1

Network 2

Network 3



Сегменты на рисунке являются частью пользовательского функционального блока . В этом блоке открывается блок DB10 и активируется функция MCR. Если выполняется безусловный вызов FC10, происходит следующее:

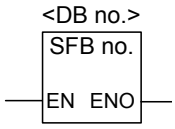
Сохраняются точка возврата в вызывающий блок и регистры блоков данных для DB10 и экземплярного блока вызываемого блока. Бит MA , установленный в 1 инструкцией MCRA сохраняется в стеке блоков BSTACK и сбрасывается в 0 для вызванного блока FC10. Выполнение программы продолжается в FC10. Если в FC10 требуется активация MCR, то это должно быть повторно сделано внутри функции. Значение RLO должно быть сохранено в бите BR с помощью инструкции [SAVE] для возможности оценки ошибки в блоке FC. После выполнения FC10, программа возвращается для выполнения в вызывающий блок. Бит MA восстанавливается, как и номер блока данных , восстанавливаемый в регистре DB. Если FC10 обработан без ошибок, состояние сигнала на выходе ENO = 1 и ,соответственно, выполняется обработка FC11, если возникали ошибки, программа продолжается со следующего сегмента. Если FC11 обработан без ошибок, состояние сигнала на выходе ENO = 1 и ,соответственно, выход Q4.0 также равен 1.

Замечание

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

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_FB выполняются следующие процедуры:

- Сохранение адреса возврата в вызывающий блок,
- Сохранение значений регистров блоков данных (DB и DI),
- Создание новой области в локальном стэке для вызываемого функционального блока.
- 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

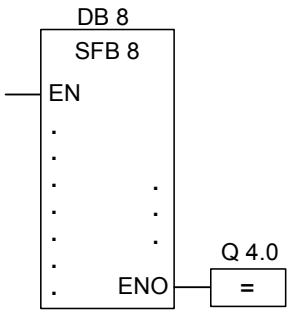
		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Условный вызов	Записывает	X	-	-	-	0	0	X	X	X
Безусловный вызов	Записывает	-	-	-	-	0	0	X	X	X

Пример

Network 1

Network 2

Network 3



Network 4

Сегменты на рисунке являются частью пользовательского функционального блока . В этом блоке открывается блок DB10 и активируется функция MCR. Если выполняется безусловный вызов SFB8, происходит следующее:

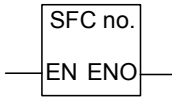
Сохраняются точка возврата в вызывающий блок и регистры блоков данных для DB10 и экземплярного блока вызывающего FB. Бит MA , установленный в 1 инструкцией MCRA сохраняется в стеке блоков BSTACK и сбрасывается в 0 для вызванного блока SFB8. Выполнение программы продолжается в SFB8. После завершения обработки 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),
- Создание новой области в локальном стеке для вызываемой системной функции.
- Бит MA (бит активности функции MCR) сохраняется в BSTACK

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

Биты слова состояния

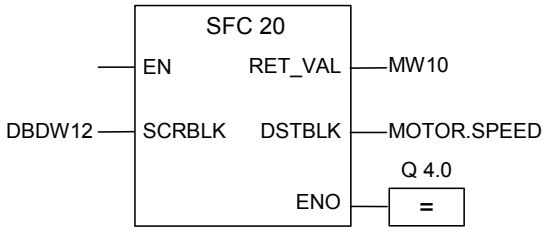
		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Условный вызов	Записывает	X	-	-	-	0	0	X	X	X
Безусловный вызов	Записывает	-	-	-	-	0	0	X	X	X

Пример

Network 1

Network 2

Network 3



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

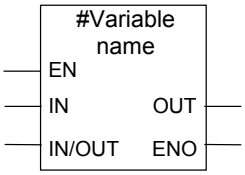
Сохраняются точка возврата в вызывающий блок и регистры блоков данных для DB10 и экземплярного блока вызывающего FB. Бит MA , установленный в 1 инструкцией MCRA сохраняется в стеке блоков BSTACK и сбрасывается в 0 для вызванного блока SFC20. Выполнение программы продолжается в SFC20. После завершения обработки SFC20 , программа возвращается в вызывающий блок. Восстанавливаются бит MA . Если SFC20 обработан без ошибок, состояние сигнала на выходе ENO = 1 и, соответственно, выход Q4.0 также равен 1.

Замечание

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

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 частого использования.

10.9 Команды Master Control Relay

Важные замечания по использованию функций MCR

Определение Master Control Relay

Master Control Relay (MCR, см. также раздел Master Control Relay Вкл./ Выкл.) - главное управляющее реле используется для активизации и деактивизации потока сигнала. Деактивированный поток сигнала соответствует последовательности команд, которая записывает нулевое значение вместо рассчитанного значения, или последовательности команд, которая оставляет неизменным существующее значение памяти. Операции, запускаемые командами, описанными ниже, зависят от MCR.

Команды **Присвоить** и **Коннектор** записывают в память 0, если MCR равно 0. Команды **Установить выход** и **Сбросить выход** оставляют существующее значение неизменным .

Следующие инструкции зависят от MCR области:

- # : Коннектор
- = : Присваивание
- R : Сброс выхода
- S : Установка выхода
- SR : SR триггер
- RS : RS триггер
- MOVE : передача значения

Инструкции, зависящие от MCR и их реакция на состояние сигнала MCR

Состояние сигнала MCR	**Присваивание, коннектор**	**Установка или сброс выхода**	**Перенос**
0 ("Выкл")	Записывает 0. (Имитирует реле, перешедшее в состояние покоя после снятия напряжения.)	Не работают. (Имитирует реле, остающееся в текущем состоянии после снятия напряжения)	Записывает 0. (Имитирует компонент, выдающий значение “0” после снятия напряжения.)
1 ("Вкл")	Нормальная работа	Нормальная работа	Нормальная работа

10.10 Важные замечания по использованию MCR функций

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

- При деактивизации MCR , значение 0 записывается всеми инструкциями присвоения в сегментах между Master Control Relay On и Master Control Relay Off. 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 On .

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, and BLOCK_FB.

Передача параметров

- Вызов с передачей актуальных параметров.

LAD/FBD

- T образные ветвления и коннекторы в Ladder или FBD языках начинаются с RLO = 0.

Способ

Устранения 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.

10.11 MCR</MCR> Включение/выключение Master Control Relay

Обозначение

Включение MCR

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

Обозначение

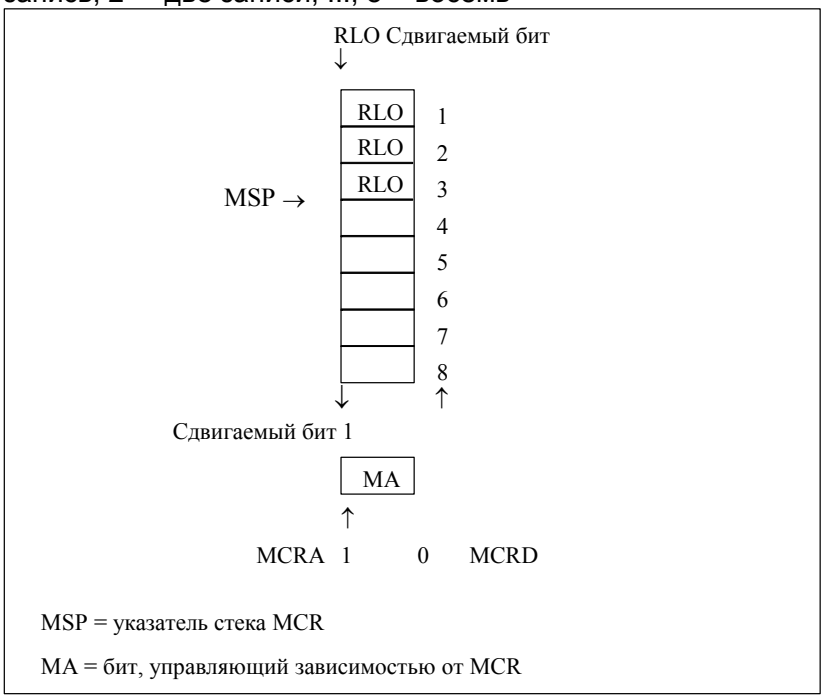
Выключение MCR

Инструкция **Выключить Master Control Relay (MCR>)** закрывает зону действия MCR, открытую последней. Эта инструкция делает это путем удаления записи RLO из стека MCR. RLO было сохранено командой **Включить Master Control Relay**. Запись, освобождаемая на другом конце LIFO-стека MCR, устанавливается в 1. Если стек уже пуст, инструкция **Выключить Master Control Relay** генерирует ошибку стека MCR (MCRF).

MCR-Stack

MCR управляется стеком шириной в один бит и глубиной восемь записей. MCR активизировано, пока все восемь записей в стеке равны 1. Инструкция MCR< копирует RLO в стек MCR. Инструкция MCR> удаляет последнюю запись из стека и устанавливает освобожденный адрес стека в 1. Если происходит ошибка, например, если в последовательности имеется более восьми команд MCR> или Вы пытаетесь выполнить инструкцию MCR>, когда стек пуст, активизируется сообщение об ошибке MCRF.

Контроль стека MCR основан на указателе стека (MSP: 0 = пуст, 1 = одна запись, 2 = две записи, ..., 8 = восемь



Инструкция **MCR<** копирует значение RLO в MCR бит.

Инструкция **MCR>** безусловно устанавливает бит MCR в 1 . Благодаря этому, все инструкции, находящиеся между MCRA и MCRD работают независимо от MCR бита.

Вложенность инструкций MCR< и MCR>

Инструкции MCR< и MCR> внутри Вашей программы всегда должны использоваться парами. Максимальная глубина вложения : 8. Другими словами , Вы можете вставить максимально 8 инструкций MCR< до вставки первой инструкции MCR>. Вы должны программировать одинаковое количество MCR< и MCR> инструкций.

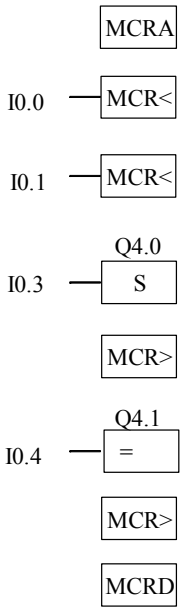
Если инструкция MCR<имеет вложение, формируется MCR бит более низкого уровня вложения. Инструкция MCR< затем сопрягает текущее состояние RLO с текущим значением бита MCR по логическому И.

После выполнения инструкции MCR> вложенного уровня, она получает бит MCR с высшего уровня.

Биты слова состояния

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

Пример



Когда команда MCRA активизирует функцию MCR, Вы можете создать до 8 вложенных зон MCR. В данном примере имеется две зоны MCR. Первая команда MCR> действует совместно с второй командой MCR<. Все команды внутри второго набора скобок MCR (MCR<MCR>) принадлежат второй зоне MCR. Операции выполняются следующим образом:

- Если I0.0 = 1: состояние сигнала на входе I0.4 присваивается выходу Q4.1.
- Если I0.0 = 0: состояние сигнала на выходе Q4.1 равно 0 независимо от состояния сигнала на входе I0.4. Выход Q4.0 остается неизменным независимо от состояния сигнала на входе I0.3.
- Если I0.0 и I0.1 = 1: выход Q4.0 устанавливается в 1, если I0.3 = 1 и Q4.1 = I0.4.
- Если I0.1 = 0: выход Q4.0 остается неизменным независимо от состояния сигнала на входе I0.3 и входе I0.0.

10.12 MCRA/MCRD : Активация/деактивация Master Control Relay

Обозначение

Активизация MCR

С помощью инструкции **Активировать Master Control Relay (главное управляющее реле)** Вы создаете последовательность команд, зависящих от MCR. После ввода этой инструкции Вы можете запрограммировать зоны MCR (см. раздел Вкл./Выкл.Master Control Relay). Когда Ваша программа активизирует область MCR, все действия MCR зависят от содержимого стека MCR

Обозначение

Деактивизация MCR

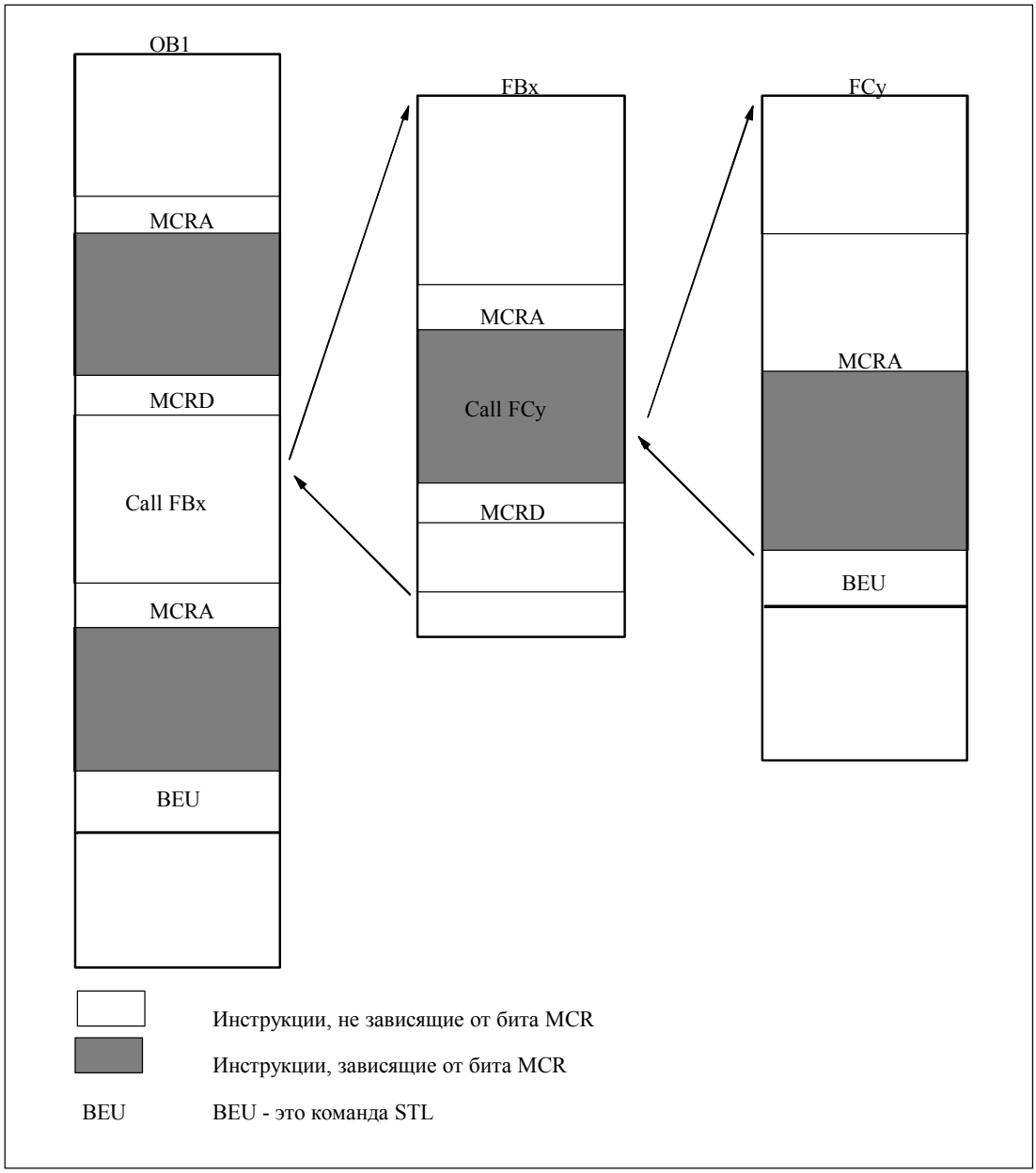
При выполнении инструкции **Деактивизировать Master Control Relay (главное управляющее реле)** последовательность команд более не зависит от MCR. После этой инструкции Вы не можете больше программировать зоны MCR. Если Ваша программа деактивизирует область MCR, то MCR всегда пропускает поток сигнала независимо от записей в стеке MCR.

Стек MCR и бит, контролирующий его зависимость (бит MA), относятся к отдельным уровням и должны быть сохранены и извлечены всякий раз, когда Вы изменяете уровень последовательности. Они предварительно устанавливаются в начале каждого уровня последовательности (входные биты MCR с 1 по 8 устанавливаются 1, указатель стека MCR устанавливается в 0, и бит MA устанавливается в 0).Стек MCR передается из блока в блок, а бит MA сохраняется и сбрасывается в «0» всякий раз, при вызове блока. Он извлекается обратно в конце блока.

MCR может быть реализован таким образом, что он оптимизирует время выполнения сгенерированного кода в CPU. Причина этого состоит в том, что последовательность команд, зависящая от MCR, не выполняется блоком; она должна быть явно активирована командой MCR. CPU распознает эту инструкцию и генерирует дополнительный код, необходимый для оценки стека MCR, пока он не распознает инструкцию MCR или не достигнет конца блока. Для команд вне диапазона MCRA/ MCRD время исполнения не увеличивается.Инструкции MCRA и MCRD всегда должны использоваться парами внутри Вашей программы.

Активация и деактивация MCR-зоны

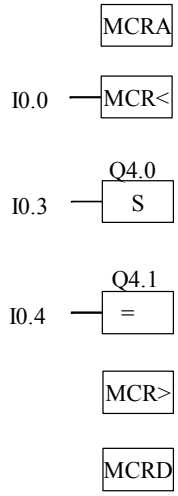
Инструкции, запрограммированные между MCRA и MCRD, зависят от состояния бита MCR. Инструкции, запрограммированные вне последовательности MCRA–MCRD, не зависят от состояния бита MCR. Если инструкция MCRD отсутствует, то инструкции, запрограммированные между инструкциями MCRA и BEU, зависят от бита MCR.



Биты слова состояния

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

Пример



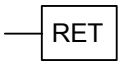
Команда MCRA активизирует функцию MCR до следующего MCRD. Команды между MCR< и MCR> обрабатываются в зависимости от бита MA (здесь I0.0):

- Если состояние сигнала на входе I0.0 равно 1:
- Выход Q4.0 устанавливается в 1, если состояние сигнала на входе I0.3 равно 1.
- Выход Q4.0 остается неизменным, если состояние сигнала на входе I0.3 равно 0.
- Состояние сигнала на входе I0.4 присваивается выходу Q4.1.
- Если состояние сигнала на входе I0.0 равно 0:
- Выход Q4.0 остается неизменным независимо от состояния сигнала на входе I0.3.
- Выход Q4.1 равен 0 независимо от состояния сигнала на входе I0.4.

Зависимость от MCR внутри FB и FC Вы должны запрограммировать самостоятельно. Если эти блоки вызываются в сегменте с активированной зоной MCRA, то не все инструкции в этом случае будут зависеть от MCR бита. Для достижения этого, используйте функцию MCRA внутри вызываемого блока.

10.13 RET: Возврат

Обозначение



Описание

Вы можете использовать инструкцию **Возврат** для выхода из блоков. Вы можете выйти из блока по условию.

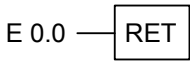
.

Биты слова состояния

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

* Инструкция **RET** внутренне состоит из последовательности "SAVE; BEC, ". Она также влияет на бит BR .

Пример



Если состояние сигнала на входе I0.0 равно 1, то обработка блока завершается.

11 Инструкции сдвига и циклического сдвига

11.1 Инструкции сдвига

11.1.1 Обзор инструкций сдвига

Описание

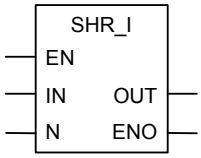
С помощью инструкций сдвига Вы можете побитно сдвигать содержимое входа IN (см. Регистры CPU) влево или вправо. Сдвиг на n битов влево умножает содержимое входа IN на 2^n; сдвиг на n битов вправо делит содержимое входа IN на 2^n. Например, если Вы сдвигаете значение 3 в двоичном коде на 3 бита влево, то получается десятичное значение 24. Если Вы сдвигаете десятичное значение 16 на 2 бита вправо, то 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 сигнала бита знака (”0” в случае положительного числа, ”1” в случае отрицательного числа). Бит, сдвигаемый последним, загружается в бит CC1 слова состояния (см. Регистры CPU). 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

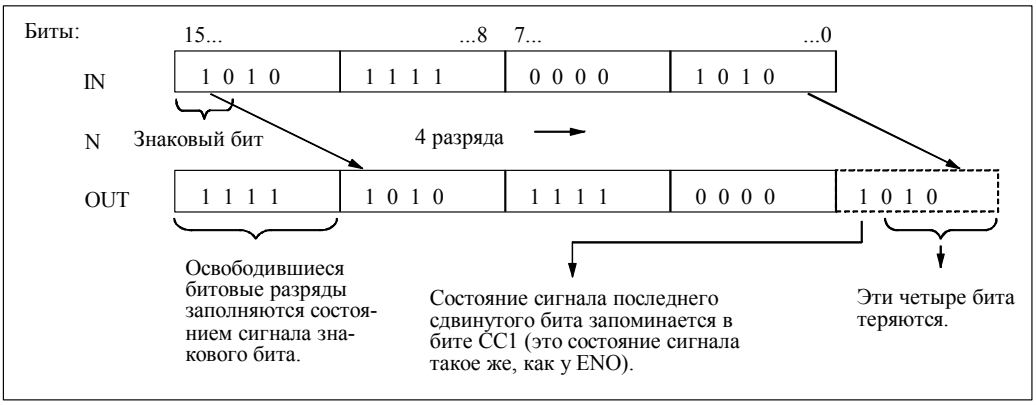


Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D, T, C	Разрешающий вход
IN	INT	I, Q, M, L, D	Сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится сдвиг
OUT	INT	I, Q, M, L, D	Результат операции сдвиг
ENO	BOOL	I, Q, M, L, D	Разрешающий выход

Описание

Инструкция **Сдвиг вправо целого числа Integer** активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно сдвигает вправо биты входа IN, имеющие номера с 0 по 15. 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.

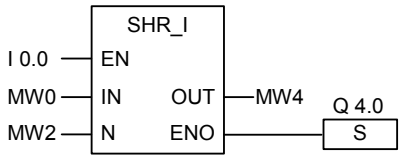
Инструкция, выполненная при не равном нулю N, всегда сбрасывает биты CC 0 и OV слова состояния в ”0”. Когда операция выполнится (EN = 1), ENO показывает состояние сигнала последнего сдвинутого бита (соответствует CC1 и VKE в слове состояния). Другие операции после этого блока, подключенные через ENO (каскадное включение), не обрабатываются, если последний сдвинутый бит имеет состояние сигнала ”0”.



Биты слова состояния

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

Пример

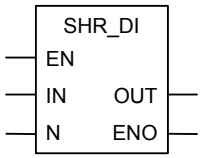


Операция активируется, если I0.0 равно 1. Меркерное слово MW0 сдвигается вправо на количество битов, заданное в MW2.

Результат сохраняется в MW4. Выход Q4.0 устанавливается, если последний сдвигаемый бит имеет состояние сигнала 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
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 сдвига
ENO	BOOL	I, Q, M, L, D	Разрешающий выход

Описание

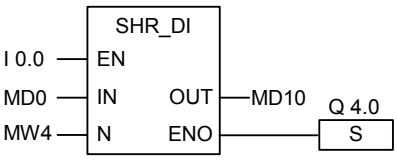
Инструкция **Сдвиг вправо числа Double Integer** активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно сдвигает вправо все содержимое входа IN. Вход N задает, на сколько битов происходит сдвиг. Если N больше, чем 32, то команда работает так, как будто N = 32. Битовые позиции слева заполняются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

Операция, запущенная при не равном нулю N, всегда сбрасывает биты A0 и OV слова состояния в ”0”. Когда операция выполнится (EN = 1), ENO показывает состояние сигнала последнего сдвинутого бита (соответствует A1 и VKE в слове состояния). Другие операции после этого блока, подключенные через ENO (каскадное включение), не обрабатываются, если последний сдвинутый бит имеет состояние сигнала ”0”.

Биты слова состояния

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

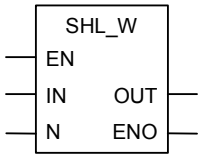
Пример



Операция активируется, если I0.0 is 1. Двойное меркерное слово MD0 сдвигается вправо на количество бит, заданное MW4.
Результат сохраняется в MD10. Выход Q4.0 устанавливается, если последний сдвигаемый бит имеет состояние сигнала 1.

11.1.4 SHL_W : Сдвиг слова влево

Обозначение

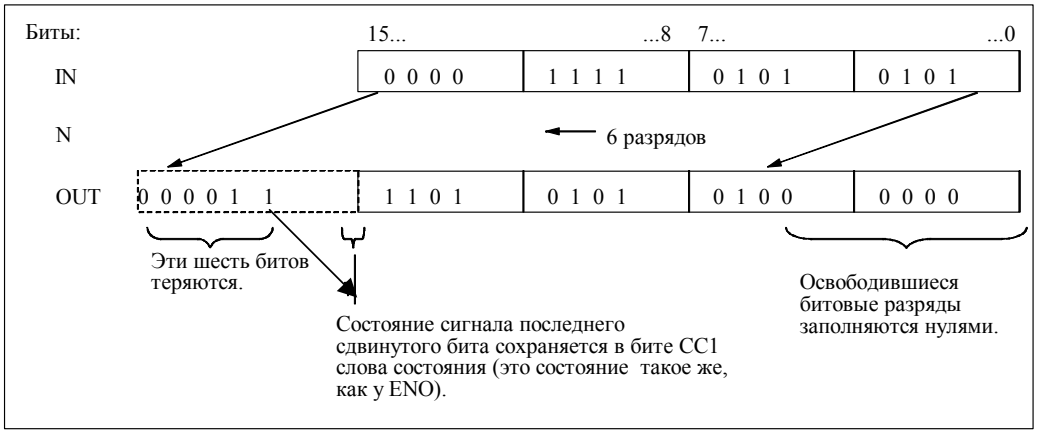


Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D, T, C	Разрешающий вход
IN	WORD	I, Q, M, L, D	Сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится сдвиг
OUT	WORD	I, Q, M, L, D	Результат операции сдвига
ENO	BOOL	I, Q, M, L, D	Разрешающий выход

Описание

Инструкция **Сдвиг слова влево** активируется , если на разрешающем входе (EN)состояние сигнала =1 и побитно сдвигает влево биты входа IN,имеющие номера с 0 по 15.

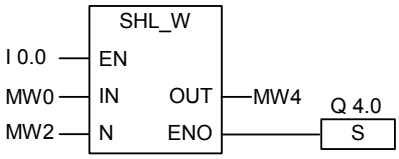
Вход N задает, на сколько бит происходит сдвиг. Если N больше, чем 16, то команда записывает 0 на выходе OUt и сбрасывает биты CC0 и OV слова состояния в ”0”. Освобождающиеся справа битовые позиции заполняются нулями. Результат операции сдвига может опрашиваться на выходе OUT. Операция, запущенная при не равном нулю N, сбрасывает биты CC0 и OV слова состояния в ”0”. Когда операция выполняется (EN = 1), ENO показывает состояние сигнала последнего сдвинутого бита (соответствует A1 и VKE в слове состояния .



Биты слова состояния

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

Пример



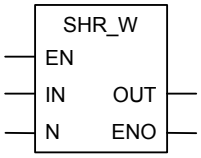
Операция активируется, сли I0.0 is 1.

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

Результат сохраняется в MW4. Выход Q4.0 устанавливается если последний сдвинутый бит имеет состояние сигнала 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
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 сдвига
ENO	BOOL	I, Q, M, L, D	Разрешающий выход

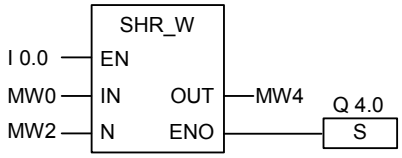
Описание

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

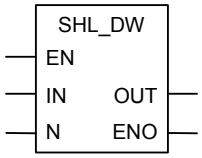
Пример



Инструкция активируется, если I0.0 равно 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
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
ENO	BOOL	I, Q, M, L, D	Разрешающий выход

Описание

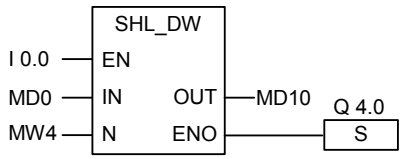
Инструкция **Сдвиг двойного слова влево** активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно сдвигает влево биты входа IN, имеющие номера с 0 по 31. Вход N задает, на сколько бит происходит сдвиг. Если N больше, чем 32, то команда записывает 0 на выходе OUT и сбрасывает биты CC0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

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

Биты слова состояния

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

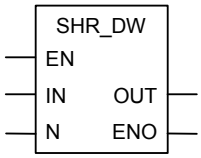
Пример



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

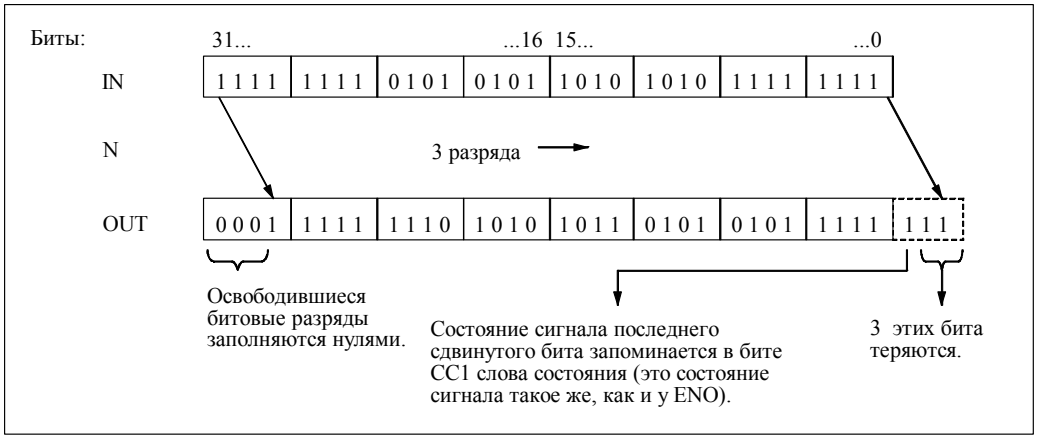


Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D, T, C	Разрешающий вход
IN	DWORD	I, Q, M, L, D	Сдвигаемое значение
N	WORD	I, Q, M, L, D	Количество битовых разрядов, на которое производится сдвиг
OUT	DWORD	I, Q, M, L, D	Результат операции сдвига
ENO	BOOL	I, Q, M, L, D	Разрешающий выход

Описание

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.

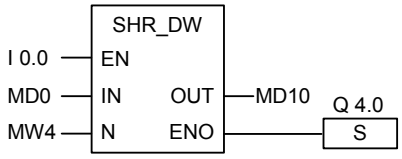
Инструкция, запущенная при не равном нулю N, всегда сбрасывает биты CC0 и OV слова состояния в ”0”. Когда инструкция выполнится (EN = 1), ENO показывает состояние сигнала последнего сдвинутого бита (соответствует CC1 и VKE в слове состояния). Другие операции после этого блока, подключенные через ENO (каскадное включение), не обрабатываются, если последний сдвинутый бит имеет состояние сигнала ”0”.



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, если последний сдвинутый бит имеет состояние сигнала 1.

11.2 Инструкции циклического сдвига

11.2.1 Обзор инструкций циклического сдвига

Описание

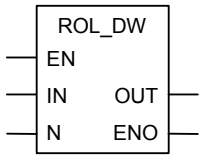
С помощью инструкций циклического сдвига, Вы можете побитно циклически сдвигать вправо или влево все содержимое входа IN. Освобождающиеся разряды заполняются состояниями сигналов тех битов, которые выталкиваются из входного значения IN. Число, задаваемое Вами для входного параметра N, показывает, на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

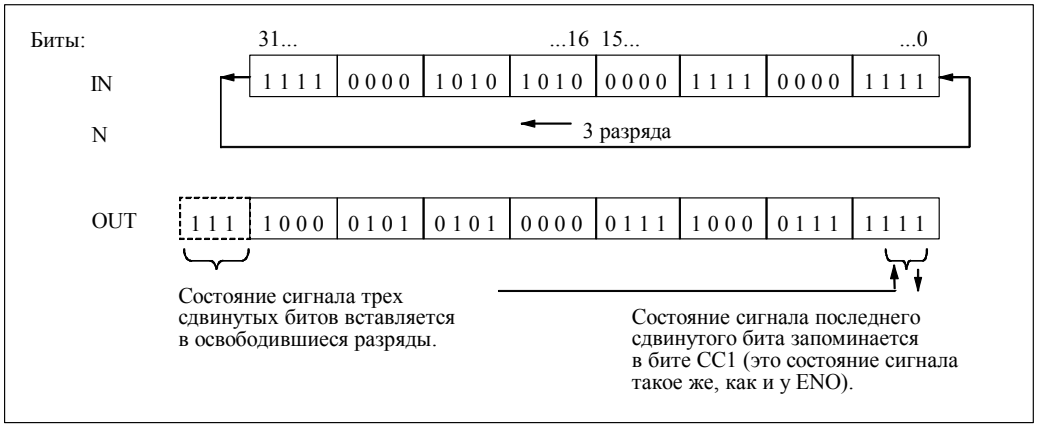
Обозначение



Параметр	Тип данных	Область памяти	Описание
EN	BOOL	I, Q, M, L, D, T, C	Разрешающий вход
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
ENO	BOOL	I, Q, M, L, D	Разрешающий выход

Описание

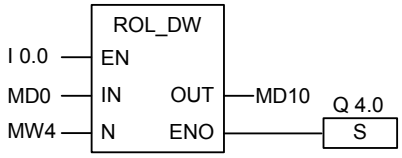
Инструкция **Циклический сдвиг двойного слова влево** активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно циклически сдвигает влево все содержимое входа IN. Вход N задает, на сколько бит происходит циклический сдвиг. Если N больше, чем 32, то двойное слово циклически сдвигается на 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”. Когда операция выполнится (EN = 1), ENO показывает состояние сигнала последнего циклически сдвинутого бита (соответствует CC1 и VKE в слове состояния).



Биты слова состояния

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

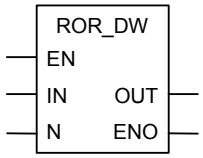
Пример



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, если последний циклически сдвинутый бит имеет состояние 1.

11.2.3 ROR_DW : Циклический сдвиг двойного слова вправо

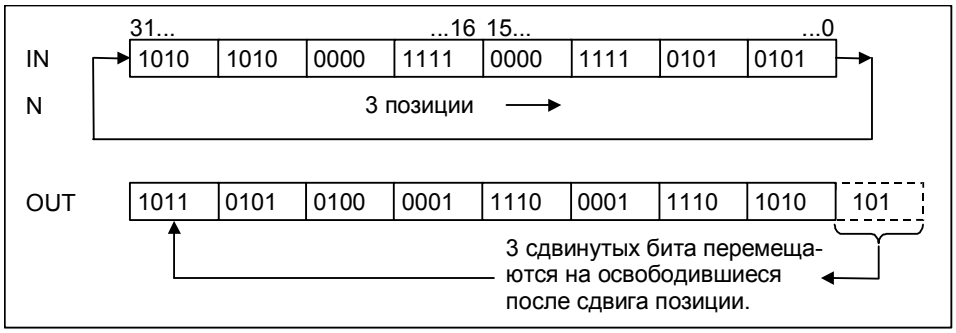
Обозначение



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

Описание

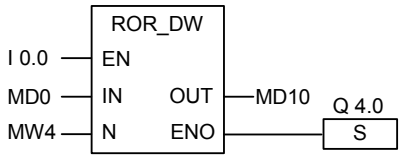
Операция **Циклический сдвиг двойного слова вправо** активируется состоянием сигнала “1” на разрешающем входе (EN) и побитно задает, на сколько битов происходит циклический сдвиг. Если N больше, чем 32, то двойное слово циклически сдвигается на число битов, равное (((N-1) по модулю 32) +1). Значение 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, если последний циклически сдвинутый бит имеет состояние сигнала 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 : “Слова состояния”. Каждая из этих инструкций реагирует на одно из следующих условий, отображаемых одним или несколькими битами слова состояния:

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

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.

Слово состояния

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

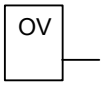
2^{15}... ...2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC

Вы можете оценивать биты слова состояния при работе с функциями:

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

12.2 OV : Бит ошибки “Переполнение”

Обозначение



Описание

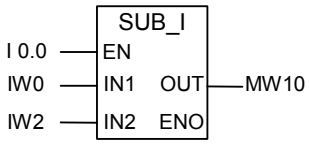
Вы можете использовать инструкцию опроса **Бита ошибки “Переполнение”** для обнаружения переполнения (OV) в последней математической операции. Если после выполнения математической операции результат оказывается за пределами допустимого диапазона в отрицательной или положительной области, то бит OV в слове состояния (см. раздел Регистры CPU) устанавливается в 1. Инструкция опрашивает состояние этого бита. Этот бит сбрасывается, если арифметическая операция была выполнена без ошибок.

Биты слова состояния

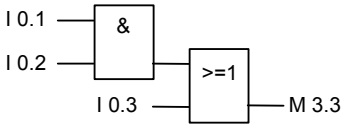
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	X	X	X	1

Пример

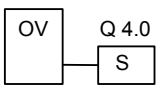
Network 1



Network 2



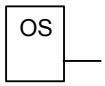
Network 3



Если состояние сигнала на входе I0.0 равно 1, то блок SUB_I активизируется. 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 статуса OV равен 1. Выход Q4.0 устанавливается, если опрос на OV равен 1 и RLO сегмента 2 равен 1 (если RLO перед выходом Q4.0 равен 1).
Если состояние сигнала на входе I0.0 равно 0 , то состояние сигнала EN и ENO равно 0.

12.3 OS : Бит ошибки “Переполнение с запоминанием”

Обозначение



Описание

В операции И эта команда комбинирует результат своего опроса с предыдущим результатом логической операции в соответствии с таблицей истинности для И . В операции ИЛИ используется таблица истинности для ИЛИ .

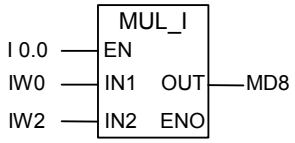
Вы можете использовать команду **Бит ошибки “Переполнение с запоминанием”** для распознавания предыдущего переполнения (OS, overflow stored - переполнение с сохранением) в математической операции. Если после ее выполнения результат оказывается за пределами допустимого диапазона в отрицательной или положительной области, то бит OS в слове состояния (см. также регистры CPU) 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 (см. раздел Бит ошибки “Переполнение с запоминанием”).

Биты слова состояния

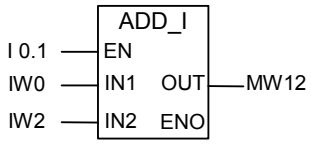
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	X	X	X	1

Пример

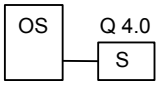
Network 1



Network 2



Network 3



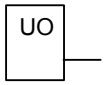
Если состояние сигнала на входе I0.0 равно 1, то активируется блок MUL_I. Если состояние сигнала на входе I0.1 равно 1, то активируется блок ADD_I. Если результат одной из этих арифметических операций выходит за пределы допустимого диапазона для целых чисел, то бит OS в слове состояния уста навливается.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

Сегмент 1: Если состояние сигнала на входе I0.0 равно 0 (не активирован), то состояние сигнала EN и ENO равно 0. Если состояние сигнала на EN равно 1 (активирован) и результат арифметической операции выходит за пределы допустимого диапазона, то состояние сигнала на ENO равно 0..

Сегмент 2: Если состояние сигнала на входе I0.1 равно 0 (не активирован), то состояние сигнала EN и ENO равно 0. Если состояние сигнала на EN равно 1 (активирован) и результат арифметической операции выходит за пределы допустимого диапазона, то состояние сигнала на ENO равно 0.

12.4 UO : Бит ошибки “Неупорядочено”

Обозначение



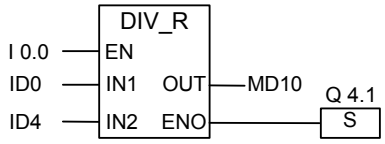
Описание

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). Для этого анализируются биты кода условия слова состояния (CC1 и CC0, см. раздел регистры CPU). Если результат арифметической операции является неупорядоченным (UO = unordered - неупорядочен), то опрос состояния сигнала дает результат 1. Если комбинация CC1 и CC0 не указывает на неупорядоченность результата арифметической операции, то опрос состояния сигнала дает 0.

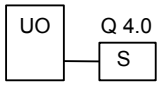
Биты слова состояния

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

Пример Network 1



Network 2



Если состояние сигнала на входе I0.0 равно 1, то блок DIV_R активизируется. Если значение одного из двойных входных слов ID0 или ID4 не является допустимым 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 Q4.0 устанавливается, если функция DIV_R выполняется, но одно из значений в арифметической функции не является допустимым числом с плавающей точкой. Если состояние сигнала на входе I0.0 равно 0 (не активирован) то состояние сигнала на EN и ENO равно 0.

12.5 BR : Бит ошибки “Двоичный результат”

Обозначение

English **German**

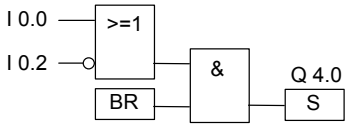
Описание

Вы можете использовать инструкции с **Битом двоичного результата** проверки статуса бита BR (Binary Result- двоичный результат) слова состояния (см. регистры CPU).

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 : Биты результата

Обозначение

== 0 —	Результат битовой инструкции **равно 0** определяет равенство выполненной математической инструкции 0.
<> 0 —	Результат битовой инструкции **не равно 0** определяет не равенство выполненной математической инструкции 0.
> 0 —	Результат битовой инструкции **больше 0** определяет больше или нет 0 результат выполненной математической инструкции.
< 0 —	Результат битовой инструкции **меньше 0** определяет меньше или нет 0 результат выполненной математической инструкции.
>= 0 —	Результат битовой инструкции **больше или равно 0** определяет больше или равен 0 результат выполненной математической инструкции.
<= 0 —	Результат битовой инструкции **меньше или равно 0** определяет меньше или равен 0 результат выполненной математической инструкции.

Описание

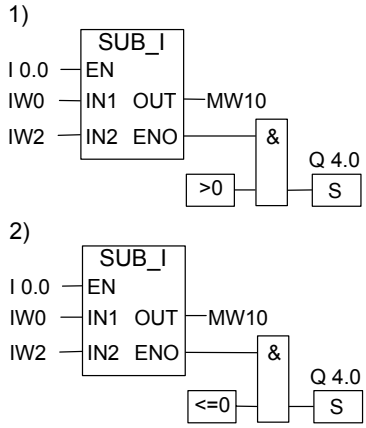
Вы можете использовать инструкции с **Битами двоичного результата** для определения отношения результата математической операции к нулю, иными словами, является ли результат ==0, <>0,>0, <0, >=0 или. <=0. Для этого анализируются биты кода условия CC1 и CC0 (см. раздел регистры CPU). Если условие сравнения, указанное в операнде, выполняется, то результат опроса этого состояния сигнала равен 1.

В операции И эта команда комбинирует результат своего опроса с предыдущим результатом логической операции (RLO) в соответствии с таблицей истинности для И . В операции ИЛИ эта команда комбинирует результат своего опроса с предыдущим результатом логической операции (RLO) 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 происходит ошибка, то состояние сигнала на ENO равно 0.

1) Выход Q4.0 устанавливается, если функция выполняется правильно и результат больше 0. Если состояние сигнала на входе I0.0 равно 0 (не активизирован), то EN и ENO равны 0.

2) Выход Q4.0 устанавливается, если функция выполняется правильно и результат не больше 0. Если состо яние сигнала на входе I0.0 равно 0 (не активизиро ван), то EN и ENO равны 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 компонентов таймера).

К области памяти таймеров имеют доступ следующие функции:

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

13.2 Области 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 FUP поддерживаются 256 таймеров. 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 единиц. 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 загрузить значение времени с использованием следующего синтаксиса:

- S5T#a**H**_bb**M**_cc**S**_ddd**MS**
  - где: a = часы, bb = минуты, cc = секунды и ddd = миллисекунды
  - База времени выбирается автоматически и значение округляется до ближайшего меньшего числа с этой базой времени.

Максимальное 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

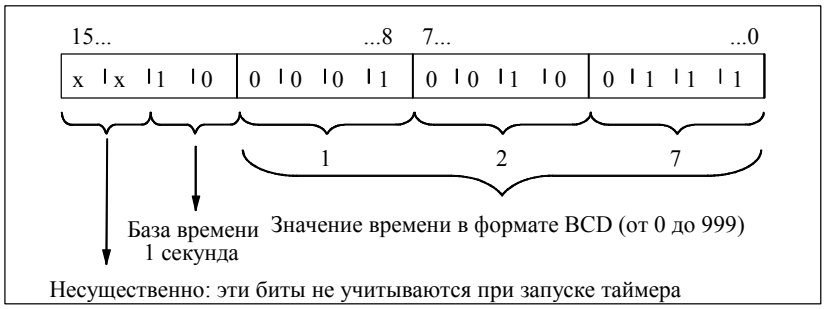
Так как значения времени запоминаются только через один интервал времени, то значения, не являющиеся точными кратными интервала времени, округляются. Значения, разрешающая способность которых слишком велика для требуемого диапазона, округляются таким образом, что достигается требуемый диапазон, но не желаемая разрешающая способность. Следующая таблица показывает возможные разрешающие способности и соответствующие диапазоны

Разрешающая способность	База времени
0.01 секунд	От 10MS до 9S_990MS
0.1 секунд	От 100MS до 1M_39S_900MS
1 секунд	От 1S до 16M_39S
10 секунд	От 10S до 2HR_46M_30S

Конфигурация битов в ячейке таймера

Когда таймер запускается, содержимое таймерной ячейки используется в качестве значения времени. Биты с 0 по 11 в таймерной ячейке содержат значение времени в двоично-десятичном формате (BCD-формат: каждая группа из четырех битов содержит двоичный код 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.

.

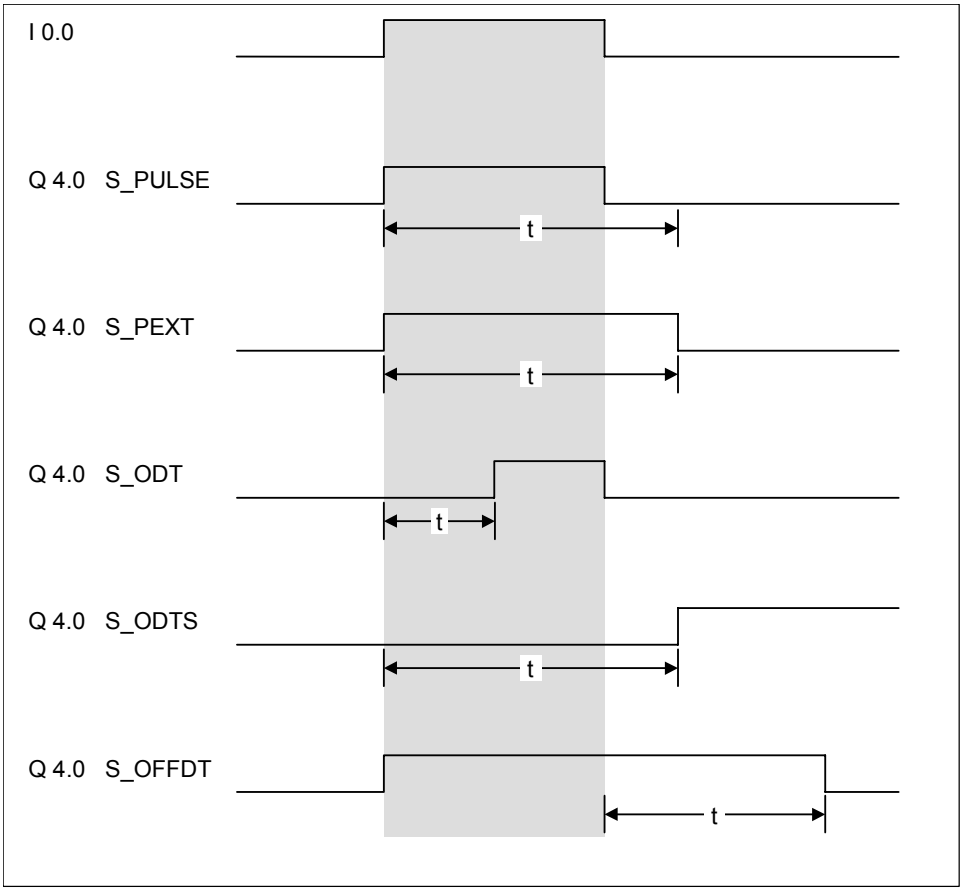


Чтение времени и базы времени

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

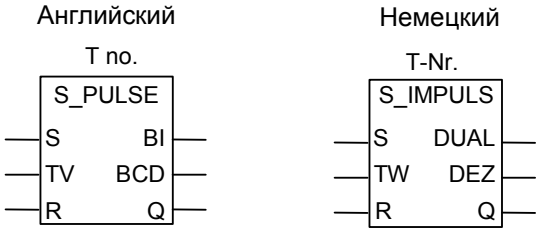
Этот рисунок помогает Вам выбрать нужный тип таймера для обработки временных событий:



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



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

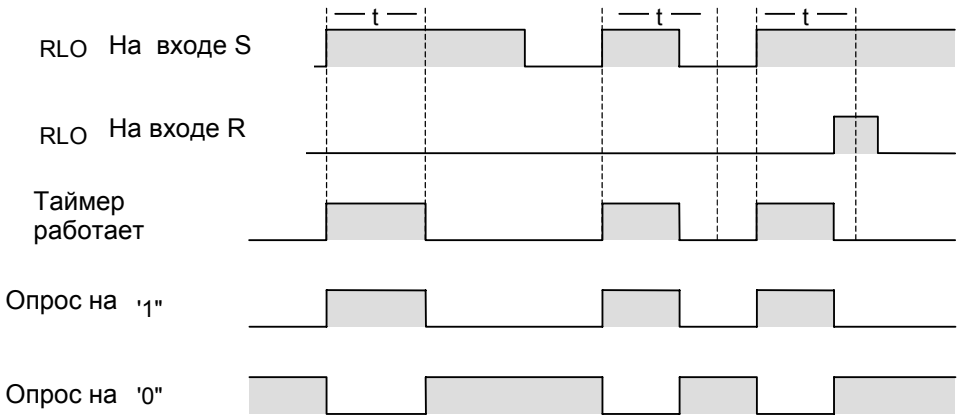
Описание

Инструкция **Задание параметров и запуск таймера «Импульс»** запускает заданный таймер, если имеется нарастающий фронт (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 состояния сигнала на 1 на выходе Q дает 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 опрошено на выходах BI и BCD. Значение времени на BI представлено в двоичном формате, а на BCD - в двоично-десятичном формате

Временные диаграммы

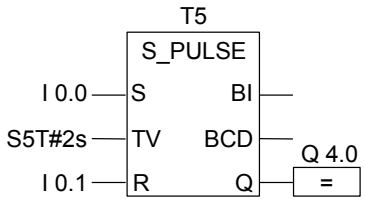
Импульсный таймер:



Биты слова состояния

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

Пример

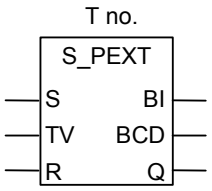


Если состояние сигнала на входе I0.0 меняется 0 на 1 (нарастающий фронт RLO), таймер T5 запускается. Таймер продолжает работать с указанным временем (2s) пока вход I0.0 равен 1. Если состояние сигнала на входе I0.0 меняется с 1 на 0 до истечения заданного времени, таймер останавливается. Если состояние сигнала на входе I0.1 меняется с 0 на 1 , когда таймер работает, то таймер сбрасывается. Состояние сигнала Q4.0 равно1 пока таймер работает.

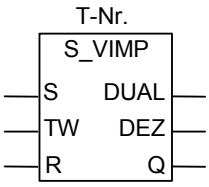
13.4 S_PEXT : Задание параметров и запуск таймера «Удлиненный импульс»

Обозначение

Английский



Немецкий



Параметры Английский	Параметры Немецкий	Тип данных	Область памяти	Описание
no.	Nr.	TIMER	T	Номер таймера. Диапазон номеров зависит от CPU.
S	S	BOOL	I, Q, M, D, L, T, C	Вход запуска
TV	TW	S5TIME	I, Q, M, D, L или константа	Установка времени(от0-9990)
R	R	BOOL	I, Q, M, D, L, T, C	Вход сброса
BI	DUAL	WORD	I, Q, M, D, L	Остаток времени (значение в целом формате)
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

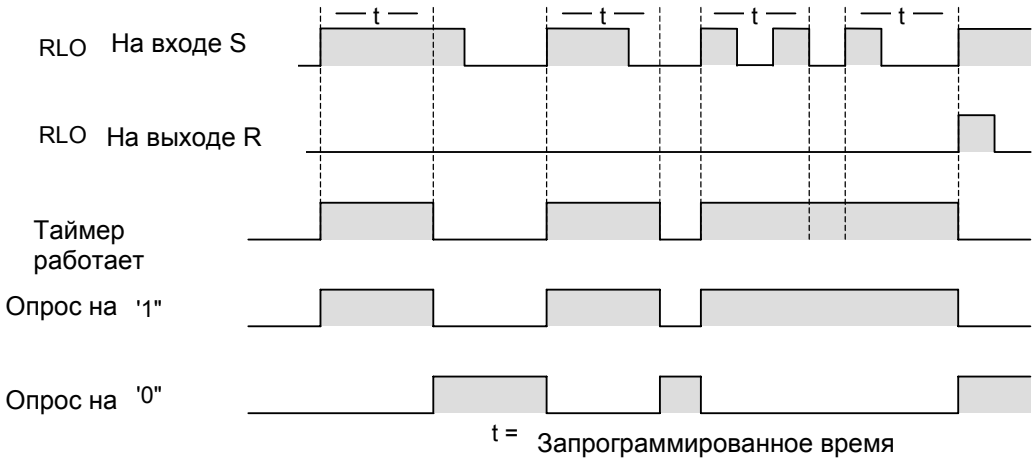
Инструкция **Задание параметров и запуск таймера «Удлиненный импульс»** запускает заданный таймер, если имеется нарастающий фронт (изменение состояния сигнала с 0 на 1) на входе запуска (S). Для запуска таймера всегда необходимо изменение сигнала. Таймер продолжает работать в течение времени, заданного на входе TV, даже если состояние сигнала на входе S меняется на 0 до истечения заданного времени. Пока таймер работает, опрос состояния сигнала на 1 на выходе Q дает 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 опрошено на выходах BI и BCD. Значение времени на BI представлено в двоичном формате, а на BCD - в двоично-десятичном формате.

Временные диаграммы

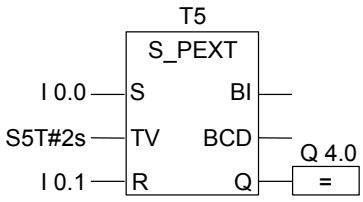
Таймер импульс с памятью:



Биты слова состояния

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

Пример

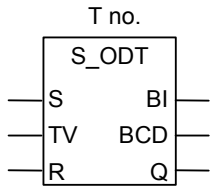


Если состояние сигнала на входе I0.0 меняется 0 на 1 (нарастающий фронт RLO), таймер T5 запускается. Таймер продолжает работать указанное время (2s) даже после сброса сигнала на входе S в 0. Если входе I0.0 вновь появляется нарастающий фронт RLO до истечения заданного времени, таймер перезапускается. 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.

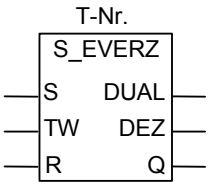
13.5 S_ODT: Задание параметров и запуск таймера «Задержка включения»

Обозначение

Английский



Немецкий



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

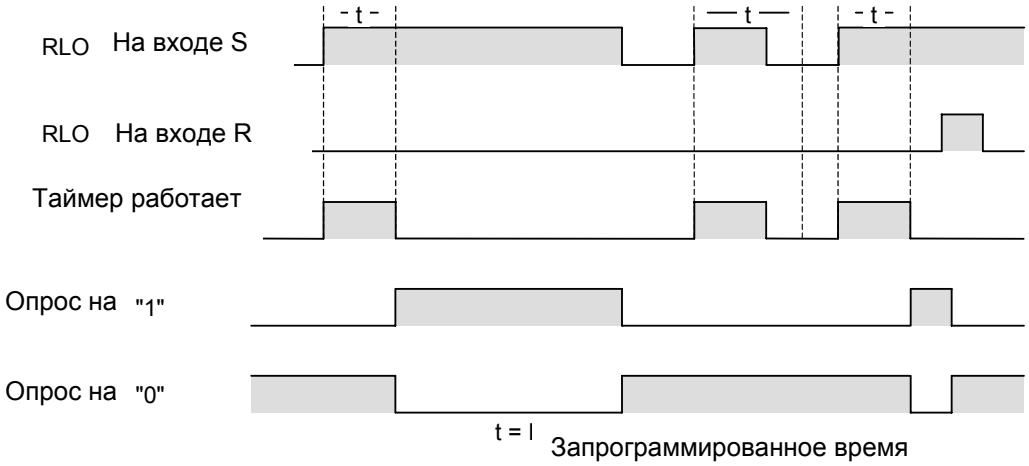
Описание

Инструкция **Задание параметров и запуск таймера «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 на выходе Q выдается 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. При этом таймер сбрасывается независимо от сигнала на входе S .

Текущее значение времени может быть опрошено на выходах BI и BCD. Значение времени на BI представлено в двоичном формате, а на BCD - в двоично-десятичном формате.

Временные диаграммы

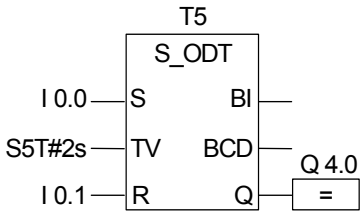
Таймер с задержкой включения:



Биты слова состояния

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

Пример

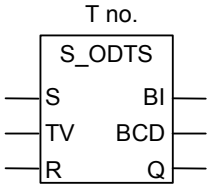


Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт в RLO), таймерT5 запускается. Если заданное время (2s) истекло и состояние сигнала на входе I0.0 равно 1, то состояние сигнала на выходе Q4.0 становится равным 1. Если состояние сигнала на входе I0.0 меняется с 1 на 0, то таймер останавливается и выход Q4.0 равен 0. Если состояние сигнала на входе S снова меняется с 0 на 1, при работе таймера, то таймер перезапускается.

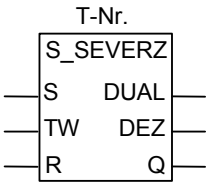
13.6 S_ODTS :Задание параметров и запуск таймера «Задержка включения с памятью»

Обозначение

Английский



Немецкий



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

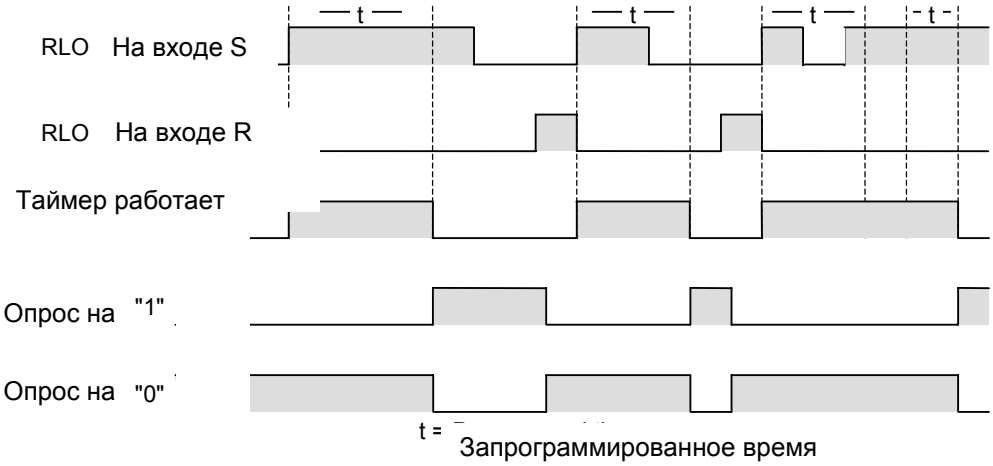
Инструкция **Задание параметров и запуск таймера «Задержка включения с памятью»** запускает заданный таймер, если имеется нарастающий фронт (изменение состояния сигнала с 0 на 1) на входе запуска (S). Для запуска таймера всегда необходимо изменение сигнала. Таймер продолжает работать в течение времени, заданного на входе TV, даже если состояние сигнала на входе S меняется на 0 до истечения заданного времени. Опрос состояния сигнала на 1 на выходе Q дает 1, когда время истекло, независимо от состояния сигнала на входе S, если вход сброса (R) остается равным нулю. 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 опрошено на выходах BI и BCD. Значение времени на BI представлено в двоичном формате, а на BCD - в двоично-десятичном формате.

Временные диаграммы

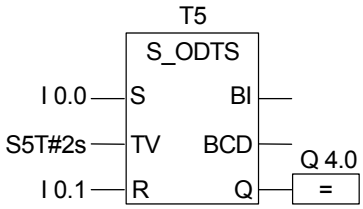
Таймер задержки включения с памятью:



Биты слова состояния

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

Пример

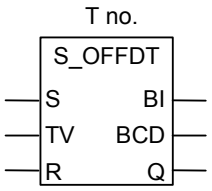


Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт в RLO), таймерT5 запускается .Таймер продолжает работать, несмотря на изменение сигнала на входе I0.0 с1 на 0. Если состояние сигнала на входе I0.0 меняется с 0 на 1 до истечения заданного времени, таймер перезапускается. Если состояние сигнала на входе I0.1 меняется с 0 на 1 во время работы таймера, то таймер сбрасывается. Состояние сигнала на выходе Q4.0 устанавливается в 1 если заданное время истекло и состояние сигнала на входе I0.1 остается равным 0.

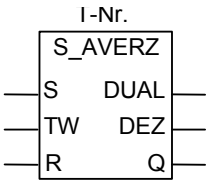
13.7 S_OFFDT: Задание параметров и запуск таймера «Задержка выключения»

Обозначение

Английский



Немецкий



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

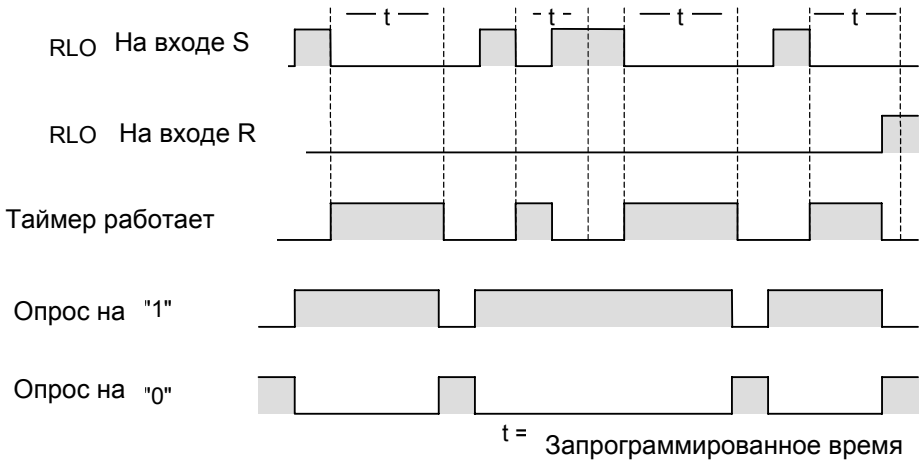
Инструкция **Задание параметров и запуск таймера «Задержка выключения»** запускает заданный таймер, если имеется падающий фронт (изменение состояния сигнала с 1 на 0) на входе запуска (S). Для запуска таймера всегда необходимо изменение сигнала. Выход Q равен 1, когда состояние сигнала на входе S равно 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 опрошено на выходах BI и BCD. Значение времени на BI представлено в двоичном формате, а на BCD - в двоично-десятичном формате

Временные диаграммы

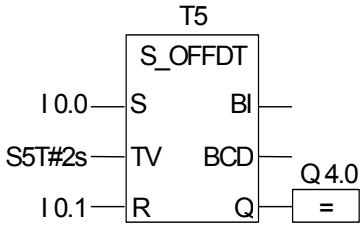
Таймер с задержкой выключения:



Биты слова состояния

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

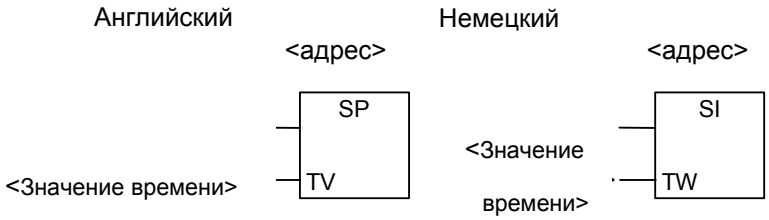
Пример



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



Параметр Английский	Параметр Немецкий	Тип данных	Область памяти	Описание
Timer no.	Timer no.	TIMER	T	Адрес определяет номер запускаемого таймера .
TV	TW	S5TIME	E, A, M, D, L или константа	Значение времени (S5TIME формат)

Описание

Инструкция **Запуск таймера «Импульс»** запускает заданный таймер, если имеется нарастающий фронт (изменение состояния сигнала с 0 на 1) на входе запуска. Пока RLO на входе запуска положительный, таймер продолжает отсчитывать заданное время. . Пока таймер работает, опрос таймера на состояние1 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.

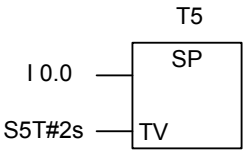
Вы можете вставлять **Запуск таймера «Импульс»** в графическом виде только на правом конце логической цепочки. Над графическим изображением элемента укажите номер таймера.

Биты слова состояния

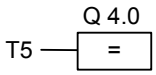
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	0	-	-	0

Пример

Network 1



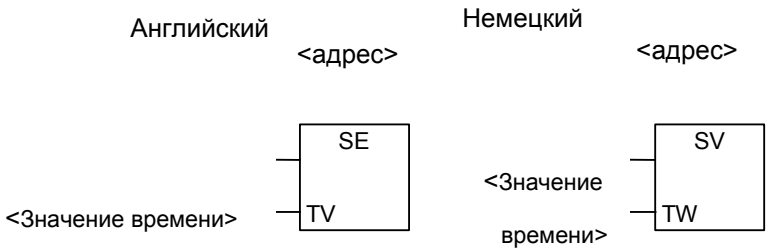
Network 2



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 , когда таймер работает, то таймер сбрасывается. Состояние сигнала Q4.0 равно1 пока таймер работает.

13.9 SE : Запуск таймера «Удлиненный импульс»

Обозначение



Параметр Английский	Параметр Немецкий	Тип данных	Область памяти	Описание
Timer no.	Timer no.	TIMER	T	Адрес определяет номер запускаемого таймера .
TV	TW	S5TIME	E, A, M, D, L или константа	Значение времени (S5TIME формат)

Описание

Инструкция **Запуск таймера «Удлиненный импульс»** запускает таймер, по нарастающему фронту (изменение состояния сигнала с 0 на 1) 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 состояние1 выдает результат 1. Таймер перезапускается с заданным временем, если состояние сигнала на входе S меняется с 0 на 1 во время работы таймера

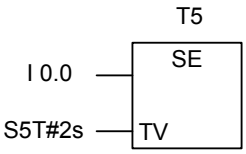
Вы можете вставлять **Запуск таймера «Удлиненный импульс»** в графическом виде только на правом конце логической цепочки. Над графическим изображением элемента укажите номер таймера.

Биты слова состояния

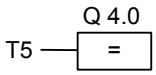
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	0	-	-	0

Пример

Network 1



Network 2

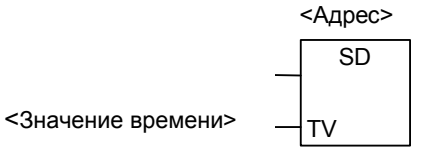


Если состояние сигнала на входе I0.0 меняется 0 на 1 (нарастающий фронт RLO), таймер T5 запускается. Таймер продолжает работать указанное время (2s) даже после сброса сигнала на входе запуска в 0. Если входе I0.0 вновь появляется нарастающий фронт RLO до истечения заданного времени, таймер перезапускается.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

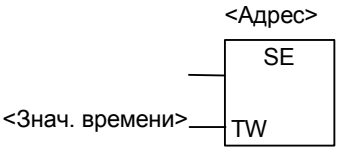
13.10 SD : Запуск таймера «Задержка включения»

Обозначение

Английский



Немецкий



Параметр Английский	Параметр Немецкий	Тип данных	Область памяти	Описание
Timer no.	Timer no.	TIMER	T	Адрес определяет номер запускаемого таймера .
TV	TW	S5TIME	E, A, M, D, L или константа	Значение времени (S5TIME формат)

Описание

Инструкция **Запуск таймера «Задержка включения»** запускает заданный таймер если имеется нарастающий фронт (изменение состояния сигнала с 0 на 1) на входе запуска . Таймер продолжает работать в течение времени, заданного на входе TV, пока состояние сигнала на входе запуска равно 1. Если состояние сигнала на входе запуска меняется с 1 на 0 во время работы таймера, таймер останавливается. В этом случае опрос таймера на состояние 1 выдает результат 0.

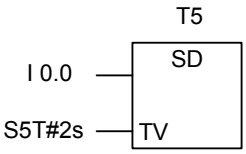
Вы можете вставлять **Запуск таймера «Задержка включения»** в графическом виде только на правом конце логической цепочки. Над графическим изображением элемента укажите номер таймера..

Биты слова состояния

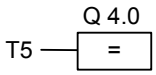
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	0	X	X	0

Пример

Network 1



Network 2

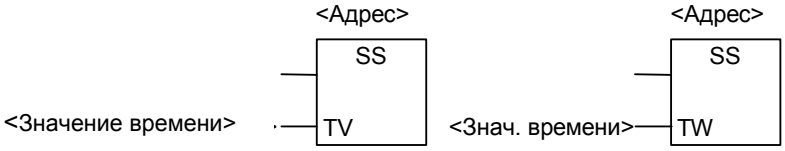


Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт RLO), таймерT5 запускается. Если заданное время (2s) истекло и состояние сигнала на входе I0.0 равно 1, то состояние сигнала на выходе Q4.0 становится равным 1. Если состояние сигнала на входе I0.0 меняется с 1 на 0, то таймер останавливается и выход Q4.0 равен 0. Если состояние сигнала на входе запуска снова меняется с 0 на 1, при работе таймера, то таймер перезапускается.

13.11 SS : Запуск таймера «Задержка включения с памятью»

Обозначение

Английский **Немецкий**



Параметр Английский	Параметр Немецкий	Тип данных	Область памяти	Описание
Timer no.	Timer no.	TIMER	T	Адрес определяет номер запускаемого таймера .
TV	TW	S5TIME	E, A, M, D, L или константа	Значение времени (S5TIME формат)

Описание

Инструкция **Запуск таймера «Задержка включения с памятью»** запускает заданный таймер, если имеется нарастающий фронт (изменение состояния сигнала с 0 на 1) 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. Опрос таймера на состояние1 дает 1, когда время истекло, независимо от состояния сигнала на входе запуска. Таймер перезапускается на заданное время, если состояние сигнала на входе запуска меняется с 0 на 1 во время работы таймера.

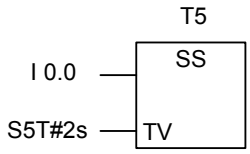
Вы можете вставлять **Запуск таймера «Задержка включения с памятью»** в графическом виде только на правом конце логической цепочки. Над графическим изображением элемента укажите номер таймера..

Биты слова состояния

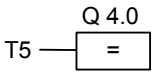
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	0	X	X	0

Пример

Network 1



Network 2

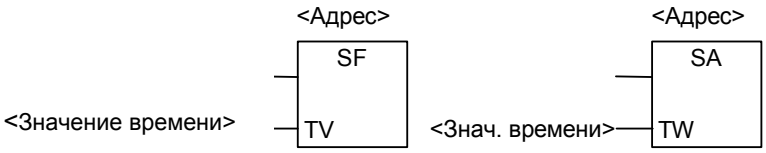


Если состояние сигнала на входе I0.0 меняется с 0 на 1 (нарастающий фронт в RLO), таймерT5 запускается .Таймер продолжает работать, несмотря на изменение сигнала на входе I0.0 с1 на 0. Если состояние сигнала на входе I0.0 меняется с 0 на 1 до истечения заданного времени, таймер перезапускается. Состояние сигнала на выходе Q4.0 устанавливается в 1 если заданное время истекло.

13.12 SF Запуск таймера «Задержка выключения»

Обозначение

Английский **Немецкий**



Параметр Английский	Параметр Немецкий	Тип данных	Область памяти	Описание
Timer no.	Timer no.	TIMER	T	Адрес определяет номер запускаемого таймера .
TV	TW	S5TIME	E, A, M, D, L или константа	Значение времени (S5TIME формат)

Описание

Инструкция **Запуск таймера «Задержка выключения»** запускает заданный таймер, если имеется падающий фронт (изменение состояния сигнала с 1 на 0) на входе запуска . Опрос таймера на состояние 1 дает результат опроса 1, когда состояние сигнала на входе запуска равно 1 или пока таймер работает. Таймер сбрасывается, когда состояние сигнала на входе запуска меняется с 0 на 1 во время работы таймера. Таймер не перезапускается, пока состояние сигнала на входе запуска снова не изменится с 1 на 0.

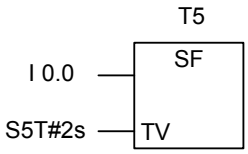
Вы можете вставлять **Запуск таймера «Задержка выключения »** в графическом виде только на правом конце логической цепочки. Над графическим изображением элемента укажите номер таймера..

Биты слова состояния

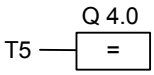
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
Записывает	-	-	-	-	-	0	-	-	0

Пример

Network 1



Network 2



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

Выход Q4.0 равен 1 когда I0.0 равен 1 или таймер работает.

14 Поразрядные логические инструкции со словами

14.1 Обзор логических инструкций со словами

Описание

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

Значение результата на выходе OUT относительно 0 влияет на бит CC в слове состояния следующим образом:

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

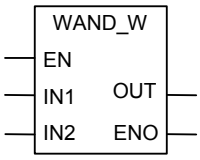
Если значение результата на выходе OUT равно 0, бит CC 1 в слове состояния сбрасывается в 0.

Для выполнения этих операций 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 со словами

Обозначение



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

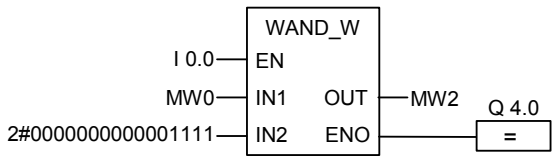
Описание

Инструкция **Поразрядное И над словами** активируется при подаче 1 на вход деблокировки (EN) .Эта команда комбинирует два слова на входах IN1 и IN2 бит за битом в соответствии с таблицей истинности для И. .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



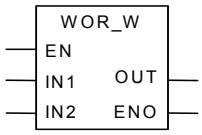
Инструкция активируется состоянием сигнала 1 на входе EN. Только биты с 0 по 3 будут сохранены, все остальные биты MW0 маскируются.

IN1 = 0101010101010101
IN2 = 0000000000001111
OUT = 0000000000000101

Q4.0 =1 если инструкция выполнялась.

14.3 WOR_W : Поразрядное ИЛИ со словами

Обозначение



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

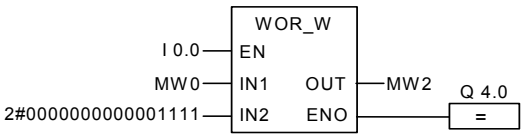
Описание

Инструкция **Поразрядное ИЛИ со словами** активируется при подаче 1 на вход деблокировки (EN) .Эта команда комбинирует два слова на входах IN1 и IN2 бит за битом в соответствии с таблицей истинности для ИЛИ. .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

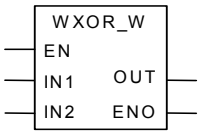


Инструкция активируется состоянием сигнала 1 на входе I0.0. Биты MW0 и константы поразрядно сопрягаются по схеме ИЛИ.При этом биты с 0 по 3 будут установлены в 1, все остальные биты MW0 останутся неизменными и будут переданы в MW2. Q4.0 =1 если инструкция выполнилась.

IN1 = 0101010101010101
IN2 = 0000000000001111
OUT = 0101010101011111

14.4 WXOR_W : Поразрядное Исключающее ИЛИ со словами

Обозначение



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

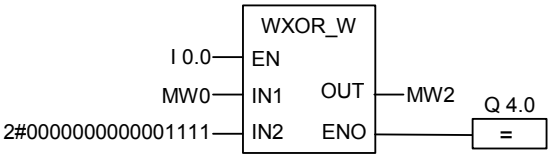
Описание

Инструкция **Поразрядное исключающее ИЛИ** со словами активируется при подаче 1 на вход деблокировки (EN) .Эта команда комбинирует два слова на входах IN1 и IN2 бит за битом в соответствии с таблицей истинности для Исключающего ИЛИ. .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



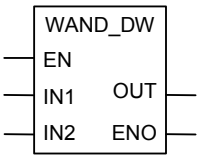
Инструкция активируется при I0.0 = 1.

IN1 = 0101010101010101
IN2 = 0000000000001111
OUT = 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, D, L, T, C	Деблокировка входа
IN1	DWORD	I, Q, M, D, L или константа	Первое значение для логической операции
IN2	DWORD	I, Q, M, D, L или константа	Второе значение для логической операции
OUT	DWORD	I, Q, M, D, L	Результат выполнения инструкции
ENO	BOOL	I, Q, M, D, L	Деблокировка выхода

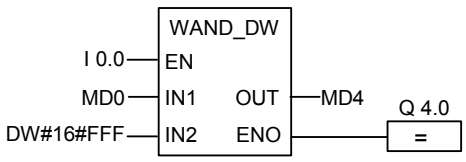
Описание

Инструкция **Поразрядное И над двойными словами** активируется при подаче 1 на вход деблокировки (EN) .Эта команда комбинирует два числа на входах IN1 и IN2 бит за битом в соответствии с таблицей истинности для логического И. .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

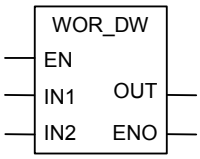


Инструкция выполняется при I0.0 = 1. Только биты с 0 по 11 сохраняются, все остальные биты MD4 маскируются. Q4.0 = 1 если инструкция выполняется.

IN1	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
OUT	=	0000000000000000	0000010101010101

14.6 WOR_DW : Поразрядное ИЛИ над двойными словами

Обозначение



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

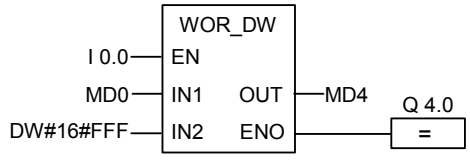
Описание

Инструкция **Поразрядное ИЛИ над двойными словами** активируется при подаче 1 на вход деблокировки (EN) .Эта команда комбинирует два числа на входах IN1 и IN2 бит за битом в соответствии с таблицей истинности для логического ИЛИ .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

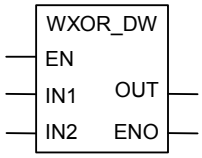


Инструкция выполняется при I0.0 = 1. Биты с 0 по 11 устанавливаются в 1, остальные передаются в MD4 без изменений. Q4.0 = 1 при выполнении функции.

IN1	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
OUT	=	0101010101010101	0101111111111111

14.7 WXOR_DW : Поразрядное Исключающее ИЛИ над двойными словами

Обозначение



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

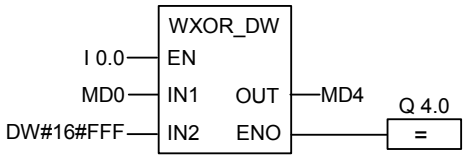
Описание

Инструкция **Поразрядное исключающее ИЛИ над двойными словами** активируется при подаче 1 на вход деблокировки (EN) .Эта команда комбинирует два числа на входах IN1 и IN2 бит за битом в соответствии с таблицей истинности для логического Исключающего ИЛИ .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. Q4.0 = 1 при выполнении инструкции.

IN1	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
OUT	=	0101010101010101	0101101010101010

A Обзор всех FBD инструкций

A.1 FBD инструкции в алфавитном порядке немецкой мнемоники (SIMATIC)

Немецкая мнемоника	Английская мнемоника	Каталог программных элементов	Описание
&	&	Bit logic instruction	Логическая операция AND(И)
>=1	>=1	Bit logic instruction	Логическая операция OR (И)
=	=	Bit logic instruction	Присвоение
#	#	Bit logic instruction	Коннектор
---|	---|	Bit logic instruction	Прямой двоичный вход
---o|	---o|	Bit logic instruction	Инверсный двоичный вход
==0	==0	Status bits	Результат сравнения
>0	>0	Status bits	Результат сравнения
>=0	>=0	Status bits	Результат сравнения
<0	<0	Status bits	Результат сравнения
<=0	<=0	Status bits	Результат сравнения
<>0	<>0	Status bits	Результат сравнения
ABS	ABS	Floating point instruction	Формирование модуля числа с плавающей точкой
ACOS	ACOS	Floating point instruction	Вычисление тригонометрических функций 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 тригонометрических функций угла для числа с плавающей точкой
ATAN	ATAN	Floating point instruction	Вычисление тригонометрических функций угла для числа с плавающей точкой
BCD_DI	BCD_DI	Convert	Преобразование BCD числа в Double Integer
BCD_I	BCD_I	Convert	Преобразование BCD числа в Integer
BIE	BR	Status bits	Бит слова состояния BR
CALL	CALL	Program control	Вызов FC/SFC без параметров
CALL_FB	CALL_FB	Program control	CALL_FB (Вызов FB в графике)
CALL_FC	CALL_FC	Program control	CALL_FC (Вызов FC в графике)
CALL_SFB	CALL_SFB	Program control	CALL_SFB (Вызов SFB в графике)
CALL_SFC	CALL_SFC	Program control	CALL_SFC (Вызов SFC в графике)
CEIL	CEIL	Convert	Преобразование числа REAL в Double Integer
CMP ? D	CMP ? D	Compare	Сравнение чисел Double Integer

Немецкая мнемоника	**Английская мнемоника**	**Каталог программных элементов**	**Описание**
CMP ? I	CMP ? I	Compare	Сравнение чисел Integer
CMP ? R	CMP ? R	Compare	Сравнение чисел Real
COS	COS	Floating point instruction	Вычисление тригонометрических функций угла для числа с плавающей точкой
DI_BCD	DI_BCD	Convert	Преобразование Integer в BCD
DI_R	DI_R	Convert	Преобразование Double Integer в Real
DIV_DI	DIV_DI	Integer math instruction	Деление чисел Double Integer
DIV_I	DIV_I	Integer math instruction	Деление чисел Integer
DIV_R	DIV_R	Floating point instruction	Деление чисел Real
EXP	EXP	Floating point instruction	Вычисление экспоненты для числа с плавающей точкой
FLOOR	FLOOR	Convert	Преобразование числа REAL в Double Integer
I_BCD	I_BCD	Convert	Преобразование Integer в BCD
I_DI	I_DI	Convert	Преобразование Integer в Double Integer
INV_I	INV_I	Convert	Инверсия числа Integer
INV_DI	INV_DI	Convert	Инверсия числа Double Integer
JMP	JMP	Jumps	Безусловный переход
JMP	JMP	Jumps	Условный переход
JMPN	JMPN	Jumps	Переход по нулю РЛО
LABEL	LABEL	Jumps	Метка перехода
LN	LN	Floating point instruction	Вычисление натурального логарифма для числа с плавающей точкой
MCR>	MCR>	Program control	Главное управляющее реле On/Off
MCR<	MCR<	Program control	Главное управляющее реле On/Off
MCRA	MCRA	Program control	Главное управляющее реле Activate/Deactivate
MCRD	MCRD	Program control	Главное управляющее реле Activate/Deactivate
MOD_DI	MOD_DI	Integer math instruction	Выделение остатка от деления
MOVE	MOVE	Move	Передача значения
MUL_DI	MUL_DI	Integer math instruction	Произведение чисел Double Integer
MUL_I	MUL_I	Integer math instruction	Произведение чисел Integer
MUL_R	MUL_R	Floating point instruction	Произведение чисел Real
N	N	Bit logic instruction	Выделение отрицательного фронта РЛО
NEG	NEG	Bit logic instruction	Выделение отрицательного фронта сигнала
NEG_DI	NEG_DI	Convert	Дополнение до двух Double Integer
NEG_I	NEG_I	Convert	Дополнение до двух Integer
NEG_R	NEG_R	Convert	Получение отрицательного числа Real
OPN	OPN	DB call	Открыть блок данных
OS	OS	Status bits	Бит переполнения с памятью
OV	OV	Status bits	Бит переполнения
P	P	Bit logic instruction	Выделение положительного фронта РЛО
POS	POS	Bit logic instruction	Выделение положительного фронта сигнала
R	R	Bit logic instruction	Сброс выхода
RET	RET	Program control	Возврат
ROL_DW	ROL_DW	Shift/Rotate	Циклический сдвиг влево Double Word

Немецкая мнемоника	Английская мнемоника	Каталог программных элементов	Описание
ROUND	ROUND	Convert	Округление числа типа Real в Double Integer
ROR_DW	ROR_DW	Shift/Rotate	Циклический сдвиг вправо Double Word
RS	RS	Bit logic instruction	RS-триггер
S	S	Bit logic instruction	Установка бита
SA	SF	Timers	Пуск таймера задержки выключения
SAVE	SAVE	Bit logic instruction	Сохранение RLO в бите BR
S_AVERZ	S_OFFDT	Timers	Задание параметров и запуск таймера задержки выключения
SE	SD	Timers	Запуск таймера задержки включения
S_EVERZ	S_ODT	Timers	Задание параметров и запуск таймера задержки включения
SHL_DW	SHL_DW	Shift/Rotate	Сдвиг влево Double Word
SHL_W	SHL_W	Shift/Rotate	Сдвиг влево Word
SHR_DI	SHR_DI	Shift/Rotate	Сдвиг вправо Double Integer
SHR_DW	SHR_DW	Shift/Rotate	Сдвиг вправо Double Word
SHR_I	SHR_I	Shift/Rotate	Сдвиг вправо Integer
SHR_W	SHR_W	Shift/Rotate	Сдвиг вправо Word
SI	SP	Timers	Запуск таймера как импульс
S_IMPULS	S_PULSE	Timers	Назначение параметров и запуск таймера как импульс
SIN	SIN	Floating point-instruction	Вычисление тригонометрических функций угла для числа с плавающей точкой
SQR	SQR	Floating point-instruction	Вычисление квадрата для числа с плавающей точкой
SQRT	SQRT	Floating point-instruction	Вычисление квадратного корня для числа с плавающей точкой
SR	SR	Bit logic instruction	SR- триггер
SS	SS	Timers	Запуск таймера задержки включения с памятью
S_SEVERZ	S_ODTS	Timers	Назначение параметров и запуск таймера задержки включения с памятью
SUB_DI	SUB_DI	Integer math-instruction	Вычитание чисел Double Integer
SUB_I	SUB_I	Integer math-instruction	Вычитание чисел Integer
SUB_R	SUB_R	Floating point-instruction	Вычитание чисел Real
SV	SE	Timers	Запуск таймера как импульс с памятью
S_VIMP	S_PEXT	Timers	Назначение параметров и запуск таймера импульс с памятью
SZ	SC	Counters	Установка значения счетчика
TAN	TAN	Floating point-instruction	Вычисление тригонометрических функций угла для числа с плавающей точкой
TRUNC	TRUNC	Convert	Выделение Double Integer из Real
UO	UO	Status bits	Опрос бита неопределенности
WAND_DW	WAND_DW	Word logic instruction	Функция побитового И над Double Word
WAND_W	WAND_W	Word logic instruction	Функция побитового И над Word
WOR_DW	WOR_DW	Word logic instruction	Функция побитового ИЛИ над Double Word
WOR_W	WOR_W	Word logic instruction	Функция побитового ИЛИ над Word

Немецкая мнемоника	**Английская мнемоника**	**Каталог программных элементов**	**Описание**
WXOR_DW	WXOR_DW	Word logic instruction	Исключающее ИЛИ над Double Word
WXOR_W	WXOR_W	Word logic instruction	Исключающее ИЛИ над Word
XOR	XOR	Bit logic instruction	Битовое исключающее ИЛИ
ZAEHLER	S_CUD	Counters	Назначение параметров и счет вверх/ вниз
ZR	CD	Counters	Счет на уменьшение
Z_RUECK	S_CD	Counters	Назначение параметров и счет вниз
ZV	CU	Counters	Счет на увеличение
Z_VORW	S_CU	Counters	Назначение параметров и счет вверх

A.2 FBD инструкции в алфавитном порядке английской мнемоники (International)

Английская мнемоника	Немецкая мнемоника	Каталог программных элементов	Описание
&	&	Bit logic instruction	Логическая операция AND(И)
>=1	>=1	Bit logic instruction	Логическая операция OR (И)
=	=	Bit logic instruction	Присвоение
#	#	Bit logic instruction	Коннектор
---|	---|	Bit logic instruction	Прямой двоичный вход
---o|	---o|	Bit logic instruction	Инверсный двоичный вход
==0	==0	Status bits	Результат сравнения
>0	>0	Status bits	Результат сравнения
>=0	>=0	Status bits	Результат сравнения
<0	<0	Status bits	Результат сравнения
<=0	<=0	Status bits	Результат сравнения
<>0	<>0	Status bits	Результат сравнения
ABS	ABS	Floating point instruction	Формирование модуля числа с плавающей точкой
ACOS	ACOS	Floating point instruction	Вычисление тригонометрических функций 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 тригонометрических функций угла для числа с плавающей точкой
ATAN	ATAN	Floating point instruction	Вычисление тригонометрических функций угла для числа с плавающей точкой
BCD_DI	BCD_DI	Convert	Преобразование BCD числа в Double Integer
BCD_I	BCD_I	Convert	Преобразование BCD числа в Integer
BR	BIE	Status bits	Бит слова состояния BR
CALL	CALL	Program control	Вызов FC/SFC без параметров
CALL_FB	CALL_FB	Program control	CALL_FB (Вызов FB в графике)
CALL_FC	CALL_FC	Program control	CALL_FC (Вызов FC в графике)
CALL_SFB	CALL_SFB	Program control	CALL_SFB (Вызов SFB в графике)
CALL_SFC	CALL_SFC	Program control	CALL_SFC (Вызов SFC в графике)
CD	ZR	Counters	Счет на уменьшение
CEIL	CEIL	Convert	Преобразование числа REAL в Double Integer
CMP ? D	CMP ? D	Compare	Сравнение чисел Double Integer
CMP ? I	CMP ? I	Compare	Сравнение чисел Integer
CMP ? R	CMP ? R	Compare	Сравнение чисел Real
COS	COS	Floating point instruction	Вычисление тригонометрических функций угла для числа с плавающей точкой
CU	ZV	Counters	Счет на увеличение
DI_BCD	DI_BCD	Convert	Преобразование Integer в BCD
DI_R	DI_R	Convert	Преобразование Double Integer в Real

Английская мнемоника	**Немецкая мнемоника**	**Каталог программных элементов**	**Описание**
DIV_DI	DIV_DI	Integer math instruction	Деление чисел Double Integer
DIV_I	DIV_I	Integer math instruction	Деление чисел Integer
DIV_R	DIV_R	Floating point instruction	Деление чисел Real
EXP	EXP	Floating point instruction	Вычисление экспоненты для числа с плавающей точкой
FLOOR	FLOOR	Convert	Преобразование числа REAL в Double Integer
I_BCD	I_BCD	Convert	Преобразование Integer в BCD
I_DI	I_DI	Convert	Преобразование Integer в Double Integer
INV_I	INV_I	Convert	Инверсия числа Integer
INV_DI	INV_DI	Convert	Инверсия числя Double Integer
JMP	JMP	Jumps	Безусловный переход
JMP	JMP	Jumps	Условный переход
JMPN	JMPN	Jumps	Переход по нулю РЛО
LABEL	LABEL	Jumps	Метка перехода
LN	LN	Floating point instruction	Вычисление натурального логарифма для числа с плавающей точкой
MCR>	MCR>	Program control	Главное управляющее реле On/Off
MCR<	MCR<	Program control	Главное управляющее реле On/Off
MCRA	MCRA	Program control	Главное управляющее реле Activate/Deactivate
MCRD	MCRD	Program control	Главное управляющее реле Activate/Deactivate
MOD_DI	MOD_DI	Integer math instruction	Выделение остатка от деления
MOVE	MOVE	Move	Передача значения
MUL_DI	MUL_DI	Integer math instruction	Произведение чисел Double Integer
MUL_I	MUL_I	Integer math instruction	Произведение чисел Integer
MUL_R	MUL_R	Floating point instruction	Произведение чисел Real
N	N	Bit logic instruction	Выделение отрицательного фронта РЛО
NEG	NEG	Bit logic instruction	Выделение отрицательного фронта сигнала
NEG_DI	NEG_DI	Convert	Дополненительный код числа Double Integer
NEG_I	NEG_I	Convert	Дополненительный код числа Integer
NEG_R	NEG_R	Convert	Получение отрицательного числа Real
OPN	OPN	DB call	Открыть блок данных
OS	OS	Status bits	Бит переполнения с памятью
OV	OV	Status bits	Бит переполнения
P	P	Bit logic instruction	Выделение положительного фронта РЛО
POS	POS	Bit logic instruction	Выделение положительного фронта сигнала
R	R	Bit logic instruction	Сброс выхода
RET	RET	Program control	Возврат
ROL_DW	ROL_DW	Shift/Rotate	Циклический сдвиг влево Double Word
ROUND	ROUND	Convert	Округление числа типа Real в Double Integer
ROR_DW	ROR_DW	Shift/Rotate	Циклический сдвиг вправо Double Word
RS	RS	Bit logic instruction	RS-триггер
S	S	Bit logic instruction	Установка выхода
SAVE	SAVE	Bit logic instruction	Сохранение RLO в бите BR

Английская мнемоника	Немецкая мнемоника	Каталог программных элементов	Описание
SC	SZ	Counters	Установка значения счетчика
S_CD	Z_RUECK	Counters	Назначение параметров и счет вниз
S_CU	Z_VORW	Counters	Назначение параметров и счет вверх
S_CUD	ZAEHLER	Counters	Назначение параметров и счет вверх/ вниз
SD	SE	Timers	Запуск таймера задержки включения
SE	SV	Timers	Запуск таймера как импульс с памятью
SF	SA	Timers	Запуск таймера задержки выключения
SHL_DW	SHL_DW	Shift/Rotate	Сдвиг влево Double Word
SHL_W	SHL_W	Shift/Rotate	Сдвиг влево Word
SHR_DI	SHR_DI	Shift/Rotate	Сдвиг вправо Double Integer
SHR_DW	SHR_DW	Shift/Rotate	Сдвиг вправо Double Word
SHR_I	SHR_I	Shift/Rotate	Сдвиг вправо Integer
SHR_W	SHR_W	Shift/Rotate	Сдвиг вправо Word
SIN	SIN	Floating point-instruction	Вычисление тригонометрических функций угла для числа с плавающей точкой
S_ODT	S_EVERZ	Timers	Назначение параметров и запуск таймера задержки включения
S_ODTS	S_SEVERZ	Timers	Назначение параметров и запуск таймера задержки включения с памятью
S_OFFDT	S_AVERZ	Timers	Задание параметров и запуск таймера задержки выключения
SP	SI	Timers	Запуск таймера как импульс
S_PEXT	S_VIMP	Timers	Назначение параметров и запуск таймера импульс с памятью
S_PULSE	S_IMPULS	Timers	Назначение параметров и запуск таймера как импульс
SQR	SQR	Floating point-instruction	Вычисление квадрата для числа с плавающей точкой
SQRT	SQRT	Floating point-instruction	Вычисление квадратного корня для числа с плавающей точкой
SR	SR	Bit logic instruction	SR- триггер
SS	SS	Timers	Запуск таймера задержки включения с памятью
SUB_DI	SUB_DI	Integer math-instruction	Вычитание чисел Double Integer
SUB_I	SUB_I	Integer math-instruction	Вычитание чисел Integer
SUB_R	SUB_R	Floating point-instruction	Вычитание чисел Real
TAN	TAN	Floating point-instruction	Вычисление тригонометрических функций угла для числа с плавающей точкой
TRUNC	TRUNC	Convert	Выделение Double Integer из Real
UO	UO	Status bits	Опрос бита неопределенности
WAND_DW	WAND_DW	Word logic instruction	Функция побитового И над Double Word
WAND_W	WAND_W	Word logic instruction	Функция побитового И над Word
WOR_DW	WOR_DW	Word logic instruction	Функция побитового ИЛИ над Double Word
WOR_W	WOR_W	Word logic instruction	Функция побитового ИЛИ над Word
WXOR_DW	WXOR_DW	Word logic instruction	Исключающее ИЛИ над Double Word
WXOR_W	WXOR_W	Word logic instruction	Исключающее ИЛИ над Word
XOR	XOR	Bit logic instruction	Битовое исключающее ИЛИ

B Примеры программирования

B.1 Обзор примеров программирования

Практические применения

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

- Управление транспортером с использованием битовых логических операций
- 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 арифметических операций с целыми числами
- Установка интервала времени для нагревания печи

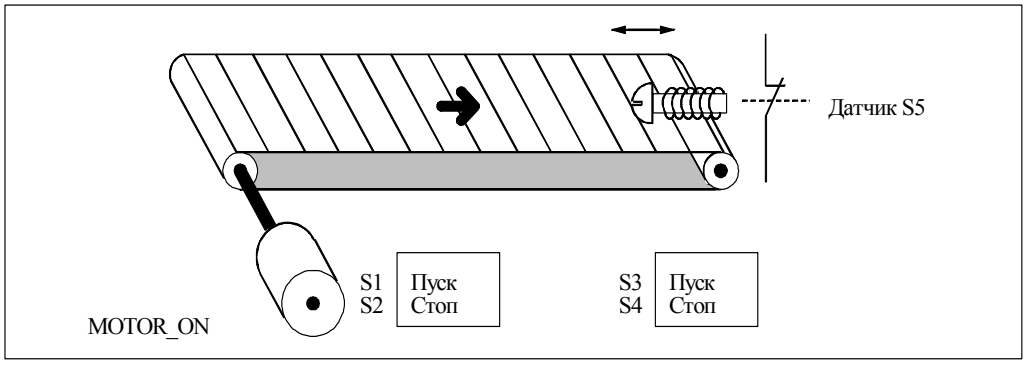
Используемые инструкции

Мнемоника	Каталог программных элементов	Описание
WAND_W	Word logic instruction	Поразрядное И со словами
WOR_W	Word logic instruction	Поразрядное ИЛИ со словами
S_CD	Counters	Обратный счет
S_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	Логическая операция И
>=1	Bit logic instruction	Логическая операция ИЛИ
=	Bit logic instruction	Присваивание
JMPN	Jumps	Переход по нулю
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.



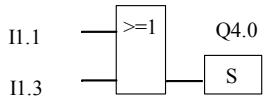
Абсолютное и символьное программирование

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).

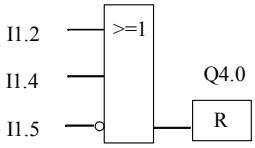
Компонент системы	Абсолютный адрес	Символ	Таблица символов
Кнопка “Пуск”	I 1.1	S1	I 1.1 S1
Кнопка “Стоп”	I 1.2	S2	I 1.2 S2
Кнопка “Пуск”	I 1.3	S3	I 1.3 S3
Кнопка “Стоп”	I 1.4	S4	I 1.4 S4
Датчик	I 1.5	S5	I 1.5 S5
Двигатель	Q 4.0	MOTOR_ON	Q 4.0 MOTOR_ON

Функциональный план управления мотором конвейера

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

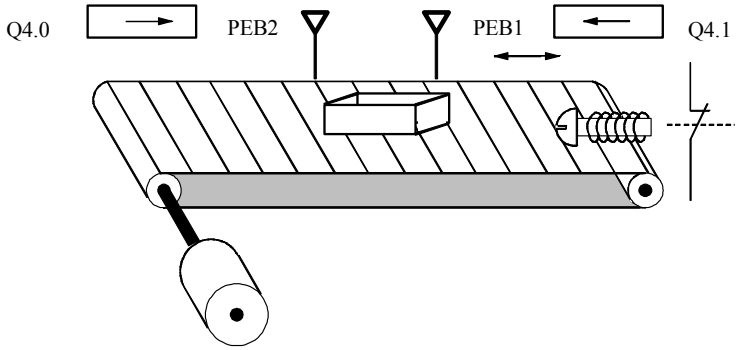


Network 2: Нажатие любой кнопки “Стоп” или срабатывание датчика вызывает отключение двигателя.



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 нормальнооткрытые контакты.



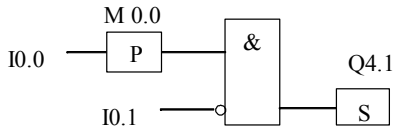
Абсолютное и символьное программирование

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).

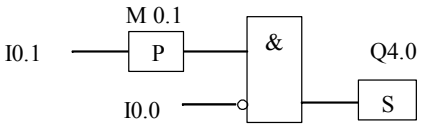
Компонент системы	Абсолютный адрес	Символ	Таблица символов
Фотоэлектрический датчик 1	I 0.0	PEB1	I 0.0 PEB1
Фотоэлектрический датчик 2	I 0.1	PEB2	I 0.1 PEB2
Индикатор движения направо	Q 4.0	RIGHT	Q 4.0 RIGHT
Индикатор движения налево	Q 4.1	LEFT	Q 4.1 LEFT
Тактовый меркер 1	M 0.0	PMB1	M 0.0 PMB1
Тактовый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

Функциональный план для определения направления движения транспортера

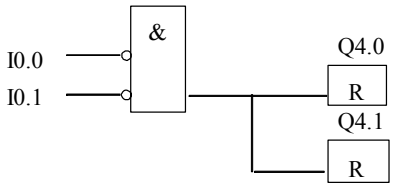
Network 1: Если на входе I0.0 сигнал изменяется с 0 на 1 (нарастающий фронт), и при этом состояние сигнала на входе I0.1 равно 0, то пакет на ленте транспортера движется влево.



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.



Network 3. Если оба фотодатчика перекрыты, то это значит, что пакет находится между датчиками.



B.3 Пример: Таймерные инструкции

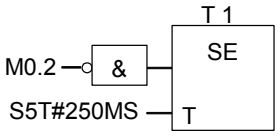
Генератор тактовых импульсов

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.

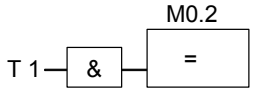
Если Вы используете S7-300, то Вы можете реализовать функцию генератора тактовых импульсов используя вызов программы из специальных организационных блоков, управляемых временем. Пример, показанный в следующей программе FBD, иллюстрирует использование таймерных функций для генерации тактовых импульсов.

Функциональный план для генератора импульсов (скважность 1:1)

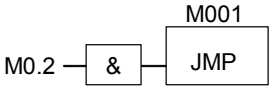
Network 1: Если статус таймера T1 равен 0, загрузить значение времени 250 мс в T1 и запустить T1 как таймер с удлиненым импульсом



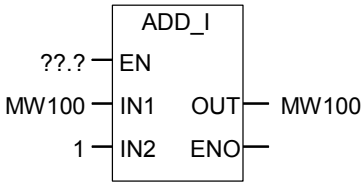
Network 2: Состояние таймера временно сохраняется во вспомогательном меркере.



Network 3: Если статус таймера T1 равен 1, перейти на метку N001.

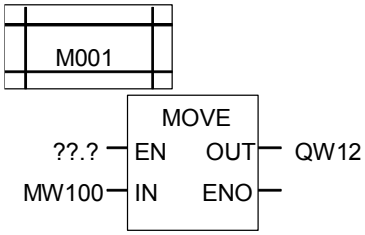


Network 4: Когда время таймера T1 истекает, меркерное слово 100 увеличивается на 1.



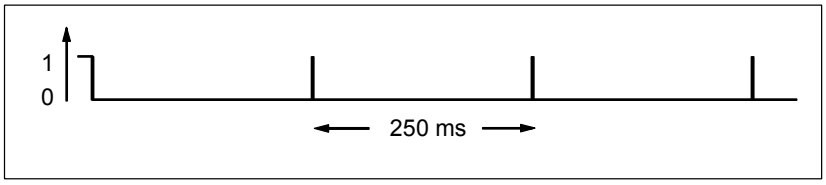
Network 5: Команда MOVE дает возможность выводить импульсы различной частоты на выходы от Q12.0 до Q13.7.

.



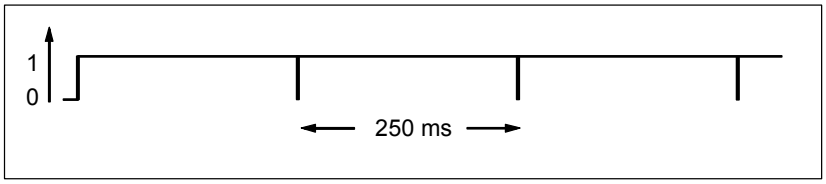
Опрос статуса

Опрос статуса таймера T1 определяет результат логической операции (RLO) инверсного входа логической операции И (AND) (M0.2 в нашем примере):



Как только время истекает, таймер перезапускается. Вследствие этого опрос статуса меркера, выполняемый командой AN M0.2, дает РЛО "1" лишь кратковременно.

Инвертированный бит РЛО:



Каждые 250 мс бит RLO равен 0. Переход игнорируется, и содержимое меркерного слова MW100 увеличивается на 1.

Получение определенной частоты

Из отдельных битов меркерных байтов MB101 и MB100 Вы можете получить следующие частоты:

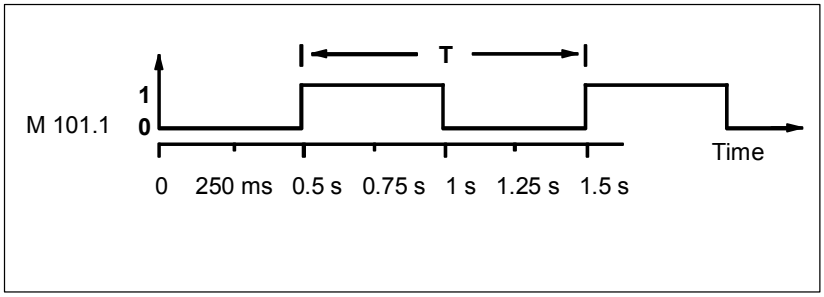
Биты в MB101/MB100	Частота в Гц	Период	
M 101.0	2.0	0.5 s	(250 ms on / 250 ms off)
M 101.1	1.0	1 s	(0.5 s on / 0.5 s off)
M 101.2	0.5	2 s	(1 s on / 1 s off)
M 101.3	0.25	4 s	(2 s on / 2 s off)
M 101.4	0.125	8 s	(4 s on / 4 s off)
M 101.5	0.0625	16 s	(8 s on / 8 s off)
M 101.6	0.03125	32 s	(16 s on / 16 s off)
M 101.7	0.015625	64 s	(32 s on / 32 s off)
M 100.0	0.0078125	128 s	(64 s on / 64 s off)
M 100.1	0.0039062	256 s	(128 s on / 128 s off)
M 100.2	0.0019531	512 s	(256 s on / 256 s off)
M 100.3	0.0009765	1024 s	(512 s on / 512 s off)
M 100.4	0.0004882	2048 s	(1024 s on / 1024 s off)
M 100.5	0.0002441	4096 s	(2048 s on / 2048 s off)
M 100.6	0.000122	8192 s	(4096 s on / 4096 s off)
M 100.7	0.000061	16384 s	(8192 s on / 8192 s off)

Состояния 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 MB101 (M101.1)

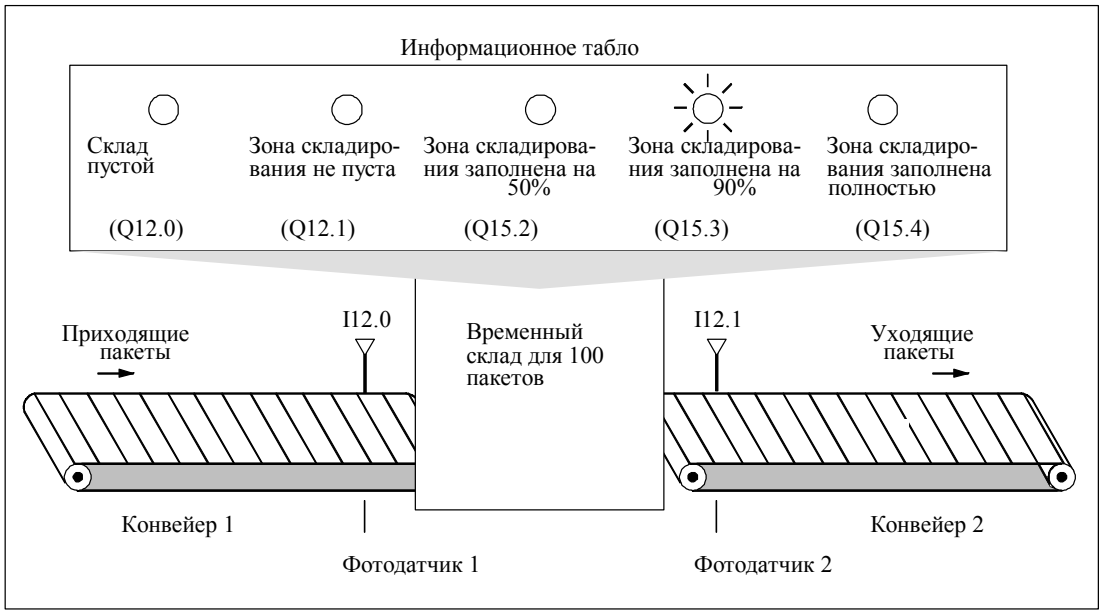
Частота = 1/T = 1/1 s = 1 Hz



B.4 Пример: Инструкции счета и сравнения

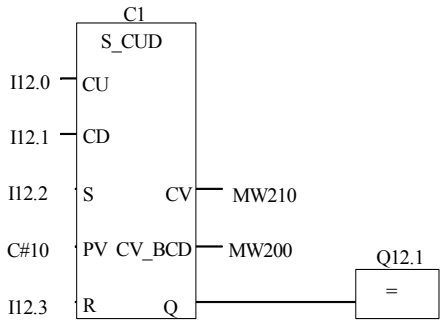
Промежуточный склад со счетчиком и компаратором

На следующем рисунке показана система с двумя конвейерами и промежуточным складом между ними. Конвейер 1 доставляет пакеты в зону складирования. Фотоэлектрический датчик в конце конвейера 1 вблизи 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.

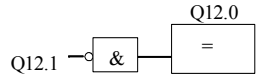


Функциональный план для работы индикации загрузки склада

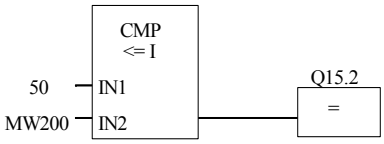
Network 1: Счетчик C1 увеличивает значение при каждом изменении сигнала с 0 на 1 на входе CU и уменьшает значение при каждом изменении сигнала с 0 на 1 на входе CD. При изменении сигнала с 0 на 1 на входе S счетчик принимает значение PV. Изменение сигнала с 0 на 1 на входе R сбрасывает значение счетчика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



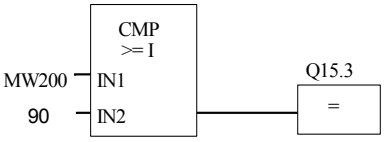
Network 2: Q12.0 показывает, что склад пуст



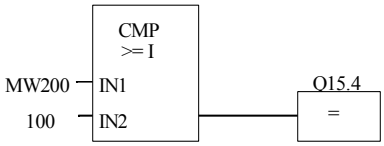
Network 3: Если 50 меньше или равно значению счетчика (иными словами, если текущее значение счетчика больше 50), то загорается индикаторная лампа “Зона складирования заполнена на 50%”



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”.



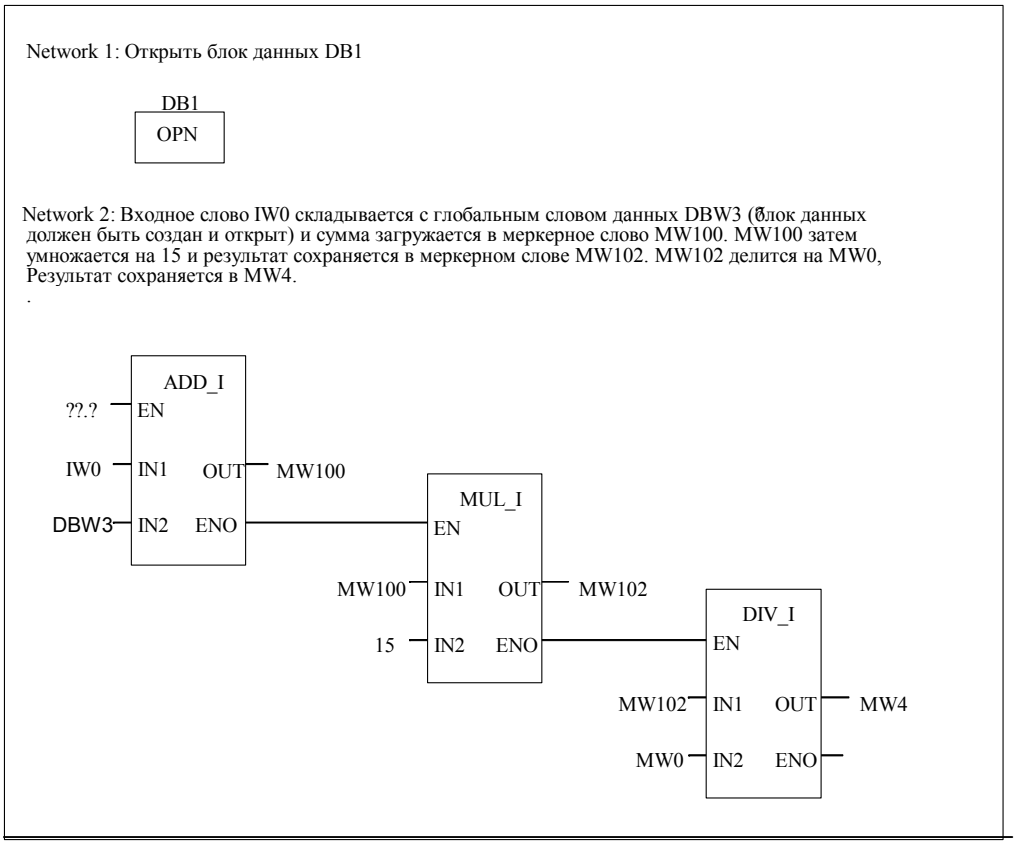
B.5 Пример: Арифметические операции с целыми числами

Решение математической задачи

Следующий пример программы показывает, как использовать арифметические операции с целыми числами и команды L и T для получения того же результата, который дает следующее уравнение:

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

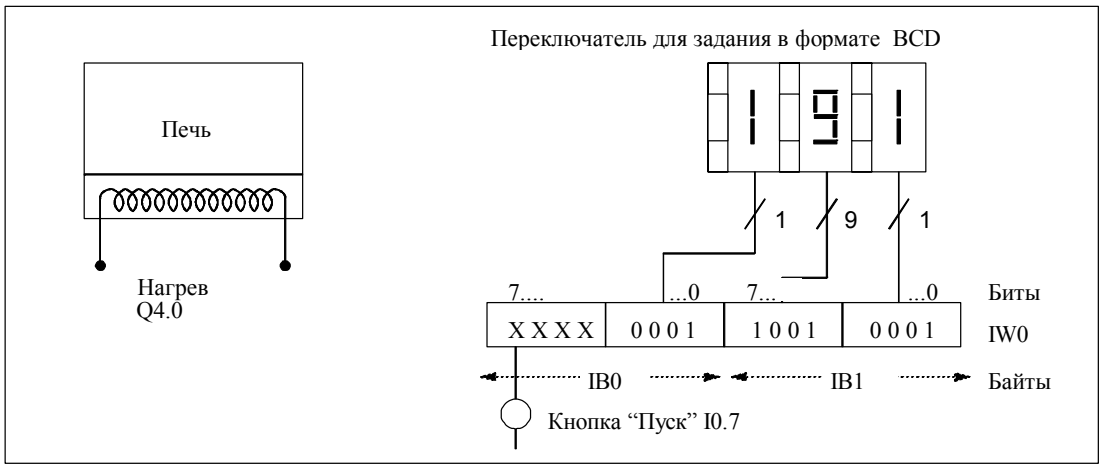
Программа в функциональном плане



B.6 Пример: Поразрядные логические операции со словами

Нагрев печи

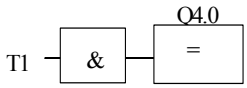
Оператор печи начинает нагревание печи нажатием кнопки “Пуск”. Оператор может устанавливать длительность нагрева с помощью цифрового переключателя, показанного на рисунке. Значение, устанавливаемое оператором задает количество секунд в двоично-десятичном формате (BCD).



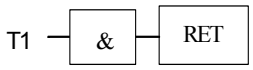
Компонент системы	Абсолютный адрес в программе FUP
Кнопка “Пуск”	I0.7
Переключатель для единиц	от I1.0 до I1.3
Переключатель для десятков	от I1.4 до I1.7
Переключатель для сотен	от I0.0 до I0.3
Запуск нагревания	Q4.0

Программа в функциональном плане

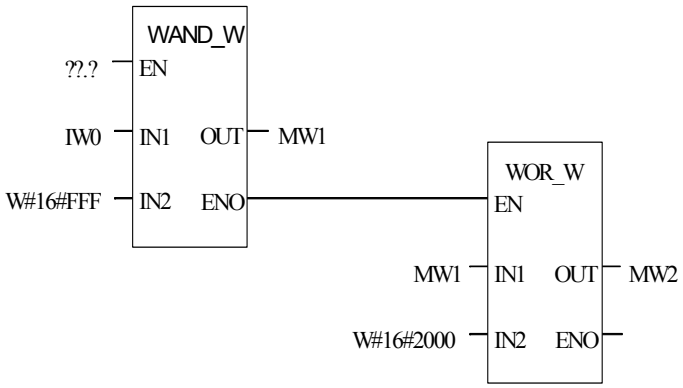
Network 1: Если таймер работает, то включается нагрев.



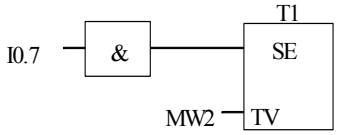
Network 2: Если таймер работает, то команда "Возврат" заканчивает обработку здесь.



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: Запустить таймер T1 в режиме удлиненого импульса, если нажата кнопка "Пуск", загружая в качестве предустановленного значения меркерное слово MW2 (полученное из предыдущих логических операций).



C Работа в функциональном плане

C.1 EN/ENO механизм

Вход разрешения (EN) и разрешение на выходе (ENO) в функциональном плане управляются при помощи бита BR .

Если EN и ENO связаны, 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.

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

1. Сложение с использованием EN и ENO заданий
2. Сложение с использованием EN задания и без ENO задания
3. Сложение без задания EN и с использованием ENO задания
4. Сложение без задания EN и без ENO задания

Замечания по созданию пользовательских блоков

При создании блоков для использования их в FBD или LAD, Вы должны обеспечить установку бита BR при выходе из блока. Четыре примера показывают что это получается не автоматически. 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 завершения 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.

C.1.1 Сложение с использованием EN и ENO заданий

При сложении с использованием EN и ENO заданий, следующие STL инструкции будут созданы:

```
1  A   I   0.0          // EN задание
2  JNB _001             // Запись RLO в BR и переход при RLO = 0
3  L     in1           // Загрузка значения параметра
4  L     in2            // Загрузка значения параметра
5         +I            // Собственно 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 A 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 использованием входа 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 значения параметра
4  L     in2           // Загрузка значения параметра
5  +I                  // Сложение
6  T     out           // Передача результата на выход
7  _001:  NOP   0
```

После выполнения инструкции на строке 1 RLO содержит результат выполненной операции. Инструкция JNB 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 с использованием выхода ENO без задания EN

В случае сложения без использованием входа EN но с заданием выхода ENO, формируется следующая STL программа :

```
1  L     in1           // Загрузка значения параметра
2        L     in2     // Загрузка значения параметра
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.1.4 Сложение без использованием входа EN и без задания ENO

В случае сложения без использованием входа EN ибез задания выхода ENO, формируется следующая STL программа:

```
1  L     in1           // Загрузка значения параметра
2  L     in2           // Загрузка значения параметра
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

Параметры блока для обработки должны получить некоторые 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 стэке вызывающего блока. Перед вызовом блока значения входных параметров копируются в экземплярный блок DB или в L -стэк. После окончания обработки блока выходные величины передаются назад в переменные. Внутри вызываемого блока Вы работаете только с копией фактичесих параметров. Инструкции STL ,необходимые для этого, создаются в вызывающем блоке и 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, а не через L-стек .

Исключение: Если входной параметр имеет тип данных BOOL , то фактическое значение, передается в блок через L -стэк.

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 -стэка.

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

Н

О

П

С

Т

У

Ц

C

F

D

E

M

S

R