OMRON

Универсальный машинный контроллер

Серия NJ/NX

Справочное руководство по командам программирования

SYSMAC always in control

W502-RU1-32

ПРИМЕЧАНИЕ

1. Все права защищены. Воспроизведение, размещение в информационно-поисковой системе или передача третьему лицу какой-либо части настоящего руководства в какой-либо форме и каким-либо способом (механическим, электронным, путем ксерокопирования, записи на носитель или иным способом) не допускается без предварительного письменного разрешения компании OMRON.
2. Использование информации, содержащейся в настоящем руководстве, не сопряжено с какой-либо патентной ответственностью.
   Кроме того, поскольку компания OMRON неуклонно стремится к совершенствованию своей продукции, информация, содержащаяся в настоящем руководстве, может быть изменена без предупреждения.
3. Подготовка настоящего руководства выполнялась с надлежащей тщательностью. Тем не менее, компания OMRON не несет ответственности за какие-либо ошибки и упущения. Компания OMRON также не несет юридической ответственности за какой-либо ущерб, возникший вследствие использования информации, содержащейся в настоящем руководстве.

Авторские права

- Экранные снимки продуктов Microsoft воспроизводятся с разрешения Microsoft Corporation.
- Данный продукт включает определенное стороннее программное обеспечение. Сведения о лицензиях и авторских правах на это программное обеспечение доступны по адресу: http://www.fa.omron.co.jp/nj_info_e/.

Введение

Благодарим вас за приобретение следующего продукта: Серия NJ/NX, модули ЦПУ.

Данное руководство содержит сведения, необходимые для использования следующего продукта: Серия NJ/NX, модули ЦПУ. Пожалуйста, прочитайте данное руководство и изучите технические характеристики и принципы работы модуля ЦПУ серии NJ/NX, прежде чем приступать к его эксплуатации в составе системы управления.
Храните данное руководство в безопасном месте, удобном для доступа во время работы.

Для кого предназначено руководство

Настоящее руководство предназначено для указанных ниже лиц, обладающих специальными знаниями в области электрических систем (инженер-электрик и т. п.).

- Персонал, ответственный за внедрение промышленных систем автоматизации.
- Персонал, ответственный за разработку промышленных систем автоматизации.
- Персонал, ответственный за установку и обслуживание промышленных систем автоматизации.
- Персонал, ответственный за администрирование оборудования промышленных систем автоматизации.

Информация о программировании, содержащаяся в данном руководстве, предназначена для персонала, понимающего спецификации языка программирования, определенные международным стандартом IEC 61131-3 или японским стандартом JIS B 3503.

Рассматриваемые продукты

В данном руководстве рассматриваются следующие продукты.

- Модули ЦПУ серии NX
  - NX701-17□□
  - NX701-16□□
  - NX102-12□□
  - NX102-11□□
  - NX102-10□□
  - NX102-90□□
  - NX1P2-11□□□□
  - NX1P2-11□□□□1
  - NX1P2-10□□□□
  - NX1P2-10□□□□1
  - NX1P2-90□□□□
  - NX1P2-90□□□□1
  - NX1P2-9B□□□□
  - NX1P2-9B□□□□1

- Модули ЦПУ серии NJ
  - NJ501-□5□□
  - NJ501-□4□□
  - NJ501-□3□□
  - NJ301-12□□
  - NJ301-11□□

- NJ101-10□□
- NJ101-90□□

Некоторые характеристики и ограничения, относящиеся к модулям ЦПУ, приводятся в других руководствах.
Обратитесь к разделам *Необходимые руководства* на стр. 3 и *Сопутствующие руководства* на стр. 32.

Необходимые руководства

В следующей таблице приведены руководства, которые имеют отношение к модулям ЦПУ серии NJ/NX. Прежде чем приступать к эксплуатации модуля ЦПУ серии NJ/NX, прочитайте все руководства, которые относятся к конфигурации вашей системы и целям ее применения. Большинство операций выполняется с помощью программного обеспечения автоматизации Sysmac Studio Automation Software. Сведения о программном обеспечении Sysmac Studio см. в руководстве *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Назначение	Руководство																			
	Основные сведения																			
	Серия NX, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NX, модуль ЦПУ NX102 — Аппаратные средства. Руководство пользователя	Серия NX, модули ЦПУ NX1P2 — Аппаратные средства. Руководство пользователя	Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя	Серия NX, модуль ЦПУ NX1P2 — Встроенные входы-выходы и дополнительная плата. Руководство пользователя	Серия NJ/NX — Команды программирования. Справочное руководство	Серия NJ/NX, модули ЦПУ — Управление движением. Руководство пользователя	Серия NJ/NX — Команды программирования для управления движением. Справочное руководство	Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя	Серия NJ/NX, модули ЦПУ — OPC UA. Руководство пользователя	Серия NX, модули ЦПУ — Протокол FINS. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Подключение к базе данных. Руководство пользователя	Серия NJ, модули ЦПУ — SECS/GEM. Руководство пользователя	Серия NJ, модули ЦПУ — Встроенные функции управления роботами. Руководство пользователя	Серия NJ, модули ЦПУ — NJ Robotics. Руководство пользователя	Серия NJ/NY, интегрированный контроллер ЧПУ — Руководство пользователя	Серия NJ/NX — Поиск и устранение неполадок. Руководство
Введение в модули ЦПУ NX701	○																			
Введение в модули ЦПУ NX102		○																		
Введение в модули ЦПУ NX1P2			○																	
Введение в контроллеры серии NJ				○																
Монтаж и настройка оборудования	○	○	○	○																
Использование функций управления движением								○												
Использование EtherCAT										○										
Использование EtherNet/IP											○									
Использование функций управления роботами для роботов OMRON																	○			

Назначение	Руководство																				
	Основные сведения																				
	Серия NX, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NX, модуль ЦПУ NX1P2 — Аппаратные средства. Руководство пользователя	Серия NX, модуль ЦПУ NX102 — Аппаратные средства. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя	Серия NX, модуль ЦПУ NX1P2 — Встроенные входы-выходы и дополнительная плата. Руководство пользователя	Серия NJ/NX — Команды программирования. Справочное руководство	Серия NJ/NX, модули ЦПУ — Управление движением. Руководство пользователя	Серия NJ/NX — Команды программирования для управления движением. Справочное руководство	Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя	Серия NJ/NX, модули ЦПУ — OPC UA. Руководство пользователя	Серия NX, модули ЦПУ — Протокол FINS. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Подключение к базе данных. Руководство пользователя	Серия NJ, модули ЦПУ — SECS/GEM. Руководство пользователя	Серия NJ, модули ЦПУ — Встроенные функции управления роботами. Руководство пользователя	Серия NJ, модули ЦПУ — NJ Robotics. Руководство пользователя	Серия NJ/NY, интегрированный контроллер ЧПУ — Руководство пользователя	Серия NJ/NX — Поиск и устранение неполадок. Руководство	
Настройка программного обеспечения					○																
Использование функций управления движением								○													
Использование EtherCAT										○											
Использование EtherNet/IP											○										
Использование OPC UA												○									
Использование FINS													○								
Использование службы подключения к базе данных														○							
Использование служб GEM															○						
Использование функций управления роботами для роботов OMRON																	○				
Использование функций управления роботами NJ Robotics																		○			
Использование функций числового программного управления																			○		
Использование функций модуля ЦПУ NX1P2						○															

Назначение	Руководство																			
	Основные сведения																			
	Серия NJ/NX — Аппаратные средства. Руководство пользователя	Серия NX, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NX, модуль ЦПУ NX102 — Аппаратные средства. Руководство пользователя	Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NX, модуль ЦПУ NX1P2 — Аппаратные средства. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Встроенные входы-выходы и дополнительная плата. Руководство пользователя	Серия NX, модуль ЦПУ NX1P2 — Встроенные входы-выходы и дополнительная плата. Руководство пользователя	Серия NJ/NX — Команды программирования. Справочное руководство	Серия NJ/NX, модули ЦПУ — Управление движением. Руководство пользователя	Серия NJ/NX — Команды программирования для управления движением. Справочное руководство	Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя	Серия NJ/NX, модули ЦПУ — OPC UA. Руководство пользователя	Серия NX, модули ЦПУ — Протокол FINS. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Подключение к базе данных. Руководство пользователя	Серия NJ, модули ЦПУ — SECS/GEM. Руководство пользователя	Серия NJ, модули ЦПУ — Встроенные функции управления роботами. Руководство пользователя	Серия NJ, модули ЦПУ — NJ Robotics. Руководство пользователя	Серия NJ, модули ЦПУ — Руководство пользователя	Серия NJ/NY, интегрированный контроллер ЧПУ — Руководство пользователя	Поиск и устранение неполадок. Руководство	Серия NJ/NX —
Создание программы пользователя								○	○											
Использование функций управления движением										○	○									
Использование EtherCAT												○								
Использование EtherNet/IP													○							
Использование OPC UA														○						
Использование FINS															○					
Использование службы подключения к базе данных																○				
Использование служб GEM																	○			
Использование функций управления роботами для роботов OMRON						○		○										○		
Использование функций управления роботами NJ Robotics																			○	
Использование функций числового программного управления																				○
Программирование обработки ошибок																				
Использование функций модуля ЦПУ NX1P2							○													

Назначение	Руководство																			
	Основные сведения																			
	Серия NX, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NX, модуль ЦПУ NX1P2 — Аппаратные средства. Руководство пользователя	Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя	Серия NX, модуль ЦПУ NX1P2 — Встроенные входы-выходы и дополнительная плата. Руководство пользователя	Серия NJ/NX — Команды программирования. Справочное руководство	Серия NJ/NX, модули ЦПУ — Управление движением. Руководство пользователя	Серия NJ/NX — Команды программирования для управления движением. Справочное руководство	Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя	Серия NJ/NX, модули ЦПУ — OPC UA. Руководство пользователя	Серия NX, модули ЦПУ — Протокол FINS. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Подключение к базе данных. Руководство пользователя	Серия NJ, модули ЦПУ — SECS/GEM. Руководство пользователя	Серия NJ, модули ЦПУ — Встроенные функции управления роботами. Руководство пользователя	Серия NJ, модули ЦПУ — NJ Robotics. Руководство пользователя	Серия NJ/NY, интегрированный контроллер ЧПУ — Руководство пользователя	Серия NJ/NX — Поиск и устранение неполадок. Руководство
Проверка работы и отладка					○															
Использование функций управления движением								○												
Использование EtherCAT										○										
Использование EtherNet/IP											○									
Использование OPC UA												○								
Использование FINS													○							
Использование службы подключения к базе данных														○						
Использование служб GEM															○					
Использование функций управления роботами для роботов OMRON																○				
Использование функций управления роботами NJ Robotics																	○			
Использование функций числового программного управления																		○		
Использование функций модуля ЦПУ NX1P2						○														
Сведения о методах обработки и устранения ошибок[*1]												△	△	△	△	△	△	△	△	○

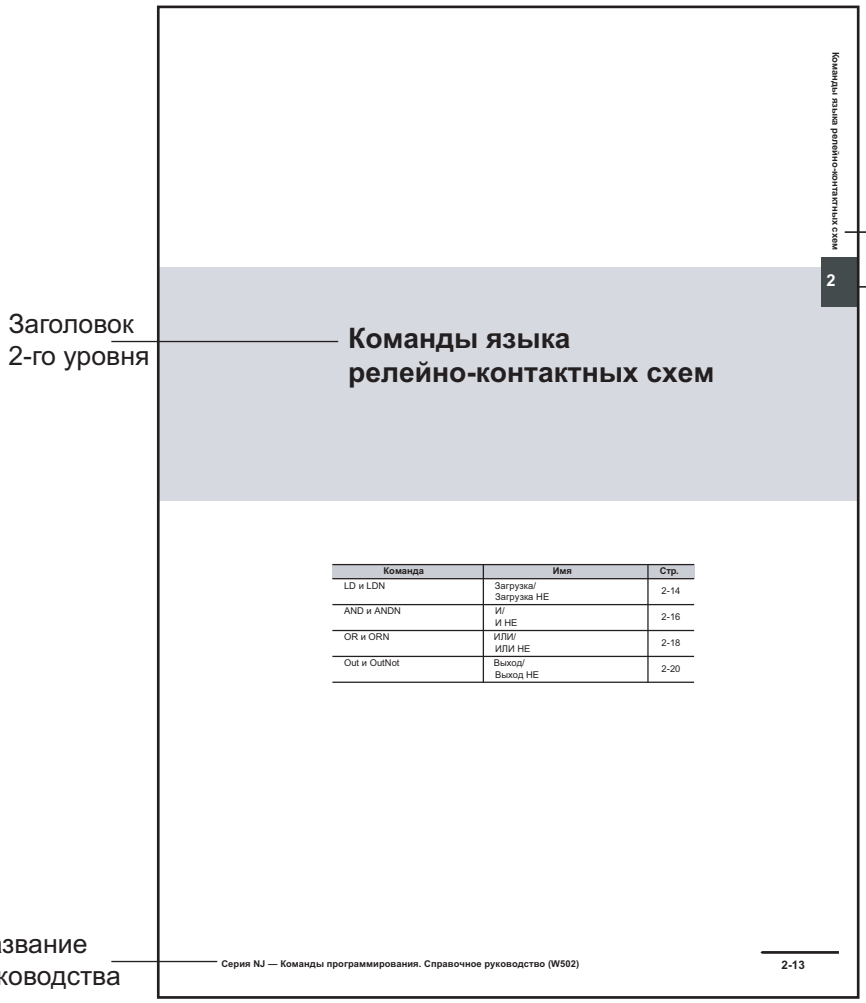
Назначение	Руководство																			
	Основные сведения																			
	Серия NX, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NX, модуль ЦПУ NX102 — Аппаратные средства. Руководство пользователя	Серия NX, модули ЦПУ NX1P2 — Аппаратные средства. Руководство пользователя	Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Встроенные входы-выходы и дополнительная плата. Руководство пользователя	Серия NX, модуль ЦПУ NX1P2 — Команды программирования. Справочное руководство	Серия NJ/NX — Команды программирования. Справочное руководство	Серия NJ/NX, модули ЦПУ — Управление движением. Руководство пользователя	Серия NJ/NX — Команды программирования для управления движением. Справочное руководство	Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя	Серия NJ/NX, модули ЦПУ — OPC UA. Руководство пользователя	Серия NX, модули ЦПУ — Протокол FINS. Руководство пользователя	Серия NJ/NX, модули ЦПУ — Подключение к базе данных. Руководство пользователя	Серия NJ/NX, модули ЦПУ — SECS/GEM. Руководство пользователя	Серия NJ, модули ЦПУ — Встроенные функции управления роботами. Руководство пользователя	Серия NJ, модули ЦПУ — NJ Robotics. Руководство пользователя	Серия NJ/NY, интегрированный контроллер ЧПУ — Руководство пользователя	Серия NJ/NX — Поиск и устранение неполадок. Руководство
Техническое обслуживание																				
Использование функций управления движением	○	○	○	○					○											
Использование EtherCAT											○									
Использование EtherNet/IP												○								

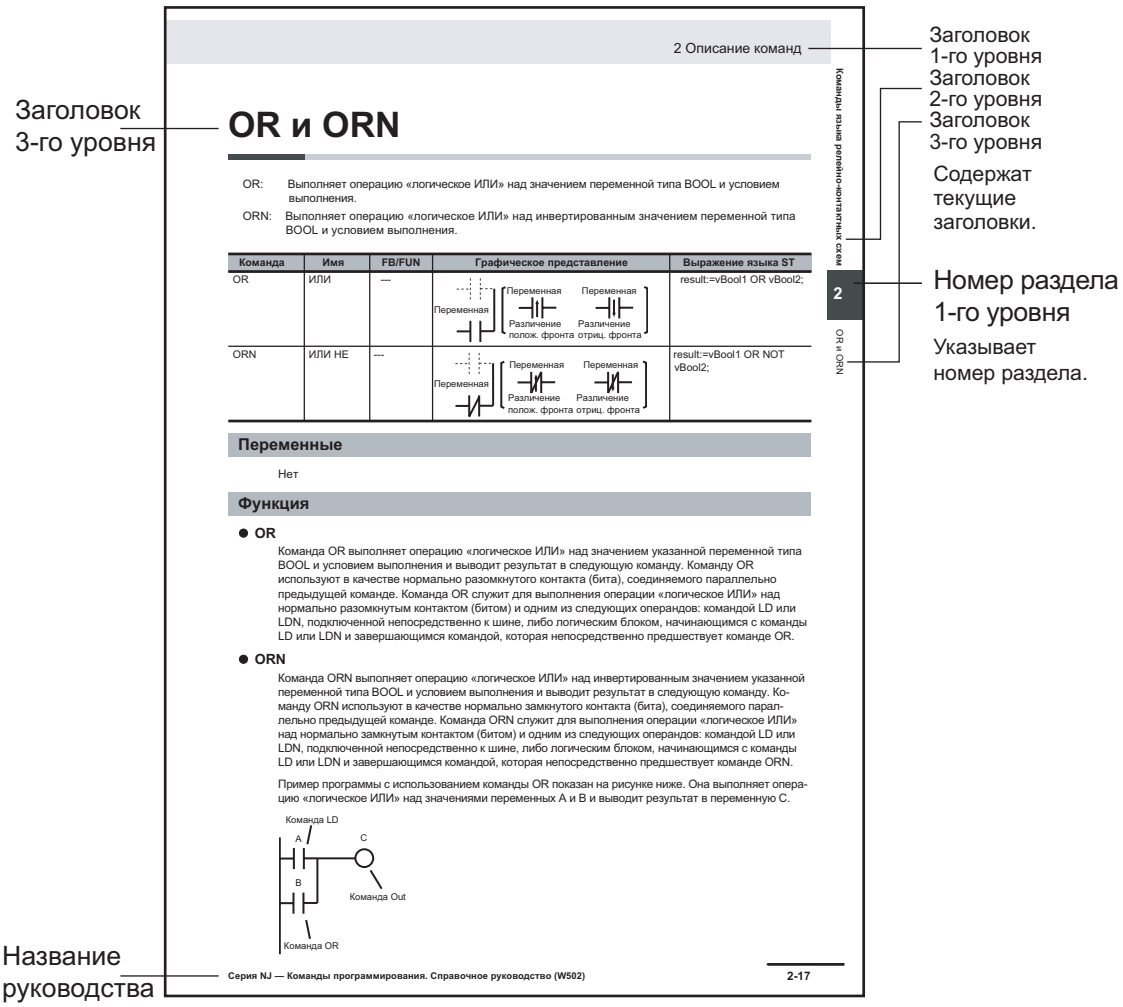
*1. Сведения о принципах обработки и устранения ошибок, а также полный список всех возможных ошибок см. в руководстве *Серия NJ/NX — Поиск и устранение неполадок. Руководство (Cat. No. W503)*. Для продуктов, руководства для которых помечены треугольником, сведения об ошибках см. в данных руководствах.

Структура руководства

Компоновка страницы

Каждая страница данного руководства имеет следующую компоновку.





OR и ORN

OR: Выполняет операцию «логическое ИЛИ» над значением переменной типа BOOL и условием выполнения.

ORN: Выполняет операцию «логическое ИЛИ» над инвертированным значением переменной типа BOOL и условием выполнения.

Команда	Имя	FB/FUN	Графическое представление	Выражение языка ST
OR	ИЛИ	---	Переменная; Переменная; Различение полож. фронта; Переменная; Различение отриц. фронта	result:=vBool1 OR vBool2;
ORN	ИЛИ НЕ	---	Переменная; Переменная; Различение полож. фронта; Переменная; Различение отриц. фронта	result:=vBool1 OR NOT vBool2;

Переменные

Нет

Функция

● **OR**

Команда OR выполняет операцию «логическое ИЛИ» над значением указанной переменной типа BOOL и условием выполнения и выводит результат в следующую команду. Команду OR используют в качестве нормально разомкнутого контакта (бита), соединяемого параллельно предыдущей команде. Команда OR служит для выполнения операции «логическое ИЛИ» над нормально разомкнутым контактом (битом) и одним из следующих операндов: командой LD или LDN, подключенной непосредственно к шине, либо логическим блоком, начинающимся с команды LD или LDN и завершающимся командой, которая непосредственно предшествует команде OR.

● **ORN**

Команда ORN выполняет операцию «логическое ИЛИ» над инвертированным значением указанной переменной типа BOOL и условием выполнения и выводит результат в следующую команду. Команду ORN используют в качестве нормально замкнутого контакта (бита), соединяемого параллельно предыдущей команде. Команда ORN служит для выполнения операции «логическое ИЛИ» над нормально замкнутым контактом (битом) и одним из следующих операндов: командой LD или LDN, подключенной непосредственно к шине, либо логическим блоком, начинающимся с команды LD или LDN и завершающимся командой, которая непосредственно предшествует команде ORN.

Пример программы с использованием команды OR показан на рисунке ниже. Она выполняет операцию «логическое ИЛИ» над значениями переменных A и B и выводит результат в переменную C.

Примечание Эти страницы показаны только в иллюстративных целях. В настоящем руководстве они могут отсутствовать или отличаться.

Особые сведения

Данное руководство содержит особые сведения следующих типов:

Меры предосторожности для обеспечения безопасной эксплуатации

Сведения об обязательных и запрещенных действиях для обеспечения безопасной эксплуатации изделия.

Меры предосторожности для обеспечения надлежащей эксплуатации

Сведения об обязательных и запрещенных действиях для обеспечения надлежащего функционирования и эксплуатационных характеристик.

Дополнительная информация

Дополнительные сведения, предоставляемые по мере необходимости.
Дополнительная поясняющая информация или информация о более простых способах выполнения тех или иных операций.

Сведения о версиях

Информация о различиях в характеристиках и функциональности для контроллеров с разными версиями модуля и для различных версий ПО Sysmac Studio.

Содержание руководства

СОДЕРЖАНИЕ

Раздел1 Набор команд

Раздел2 Описание команд

Приложения

Предметный указатель

Условия и ограничения

Гарантийные обязательства и ограничение ответственности

Гарантийные обязательства

- **Исключительная гарантия**

  Компания Omron дает исключительную гарантию того, что в течение двенадцати месяцев (или иного периода, указанного компанией Omron в письменной форме) с даты продажи изделия компанией Omron в изделии будут отсутствовать дефекты, связанные с материалами и качеством изготовления изделия. Компания Omron не признает какие-либо иные явно выраженные или подразумеваемые гарантийные обязательства.

- **Ограничения**

  КОМПАНИЯ OMRON НЕ ДАЕТ НИКАКИХ ГАРАНТИЙ ИЛИ ОБЯЗАТЕЛЬСТВ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, В ОТНОШЕНИИ НЕНАРУШЕНИЯ ПРАВ ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ, СООТВЕТСТВИЯ ИЗДЕЛИЙ ОЖИДАНИЯМ ПОКУПАТЕЛЯ И ПРИГОДНОСТИ ИЗДЕЛИЙ ДЛЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ. ПОКУПАТЕЛЬ ПРИЗНАЕТ, ЧТО ОПРЕДЕЛЕНИЕ СООТВЕТСТВИЯ ИЗДЕЛИЙ ТРЕБОВАНИЯМ, ПРЕДЪЯВЛЯЕМЫМ ПОКУПАТЕЛЕМ, НАХОДИТСЯ В КОМПЕТЕНЦИИ САМОГО ПОКУПАТЕЛЯ.

  Компания Omron также отказывается от каких-либо гарантийных обязательств и от любого вида ответственности в отношении претензий или расходов, возникших в результате нарушения прав третьих лиц и любых прав интеллектуальной собственности, тем или иным образом связанного с изделиями.

- **Удовлетворение претензий покупателя**

  Единственным обязательством компании Omron по настоящему соглашению является выполнение компанией Omron одного из следующих действий по своему выбору: (1) замена ненадлежащего изделия (путем поставки изделия в его первоначальном виде без компенсации покупателю связанной с этим стоимости трудозатрат на демонтаж или повторный монтаж изделия), (2) ремонт ненадлежащего изделия или (3) возврат покупателю или принятие к зачету денежных средств в сумме, равной цене приобретения ненадлежащего изделия; при условии, что ни при каких обстоятельствах компания Omron не будет нести ответственности по связанным с изделиями гарантийным обязательствам, ремонту, возмещению вреда или любым другим искам или расходам, если в результате анализа, проведенного компанией Omron, будет установлено, что в отношении изделий нарушались правила эксплуатации, хранения, монтажа и технического обслуживания, что изделия подвергались загрязнению, либо использовались не по назначению или подвергались недопустимой модификации или ремонту. Перед возвратом любых изделий покупатель должен получить письменное согласие компании Omron. Компания Omron, включая любые ее филиалы, дочерние компании и подразделения (далее совместно именуемые как «Компания Omron»), не несет ответственности за пригодность либо непригодность изделий для использования в комбинации с какими-либо электрическими или электронными элементами, схемами, сборочными узлами, в сочетании с какими-либо материалами, веществами или средами, а также не несет ответственности за

результаты такого использования. Никакие советы, рекомендации или сведения, предоставленные в устной или письменной форме, не могут рассматриваться в качестве дополнения или поправки к изложенным выше гарантийным обязательствам.

Официально публикуемую информацию можно найти на веб-сайте http://www.omron.com/global/ или получить у регионального представителя компании Omron.

Ограничение ответственности и др.

КОМПАНИЯ OMRON НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА КОСВЕННЫЕ, СЛУЧАЙНЫЕ, ПОБОЧНЫЕ ИЛИ ФАКТИЧЕСКИЕ УБЫТКИ, УПУЩЕННУЮ ВЫГОДУ ИЛИ КОММЕРЧЕСКИЕ ПОТЕРИ, КАКИМ БЫ ТО НИ БЫЛО ОБРАЗОМ СВЯЗАННЫЕ С ИЗДЕЛИЯМИ, НЕЗАВИСИМО ОТ ТОГО, ПРЕДЪЯВЛЯЕТСЯ ЛИ ИСК НА ОСНОВАНИИ КОНТРАКТА, ГАРАНТИЙНЫХ ОБЯЗАТЕЛЬСТВ, В СВЯЗИ С ДОПУЩЕННОЙ НЕБРЕЖНОСТЬЮ ИЛИ НА ОСНОВАНИИ БЕЗУСЛОВНОГО ОБЯЗАТЕЛЬСТВА.

Кроме того, ни при каких обстоятельствах ответственность компании Omron не может превысить собственную стоимость изделия, на которое распространяется ответственность компании Omron.

Замечания по применению

Пригодность для конкретного применения

Компания Omron не несет ответственности за соответствие каким-либо стандартам, нормативам или правилам, которые действуют в каждом конкретном случае использования изделия или его применения в составе оборудования покупателя. По запросу покупателя компания Omron предоставит соответствующую сертификационную документацию, выданную сторонними организациями, в которой указываются обеспечиваемые номинальные параметры и ограничения на применение изделия. Однако сама по себе эта информация не является достаточной для полного установления пригодности изделия для применения в конечном изделии, машине, оборудовании, системе или в других областях и целях применения. Определение пригодности конкретного изделия для применения в конечном изделии, системе или в иных целях является обязанностью исключительно самого покупателя. Ответственность за применение изделия во всех случаях несет покупатель.

НИ В КОЕМ СЛУЧАЕ НЕ ИСПОЛЬЗУЙТЕ ИЗДЕЛИЕ В БОЛЬШИХ КОЛИЧЕСТВАХ ЛИБО ДЛЯ ЦЕЛЕЙ, ПРЕДПОЛАГАЮЩИХ СЕРЬЕЗНУЮ УГРОЗУ ДЛЯ ЖИЗНИ ИЛИ ИМУЩЕСТВА ЛЮДЕЙ, НЕ УБЕДИВШИСЬ ПРЕДВАРИТЕЛЬНО В ТОМ, ЧТО БЕЗОПАСНОСТЬ ОБЕСПЕЧЕНА ВО ВСЕЙ СИСТЕМЕ В ЦЕЛОМ, А ТАКЖЕ В ТОМ, ЧТО ИЗДЕЛИЕ(-Я) OMRON ИМЕЮТ НАДЛЕЖАЩИЕ НОМИНАЛЬНЫЕ ХАРАКТЕРИСТИКИ И НАДЛЕЖАЩИМ ОБРАЗОМ СМОНТИРОВАНЫ В СООТВЕТСТВИИ С ЦЕЛЬЮ ПРИМЕНЕНИЯ ВО ВСЕЙ СИСТЕМЕ ИЛИ ОБОРУДОВАНИИ.

Программируемые изделия

Компания Omron не несет ответственности за программы пользователя, создаваемые для программируемых изделий, а также за какие-либо последствия, возникшие в результате их применения.

Отказ от ответственности

Технические данные

Технические данные, представленные на веб-сайтах, в каталогах и других материалах компании Omron, не являются предметом гарантийного обязательства и предназначены исключительно для определения пригодности изделий для нужд пользователей. Эти данные могут соответствовать определенным условиям, при которых производились испытания компанией Omron, и пользователи должны соотносить их с фактическими предстоящими условиями эксплуатации изделий. Предметом «Гарантийных обязательств и ограничения ответственности» являются характеристики с учетом фактических условий эксплуатации.

Изменение характеристик

Характеристики изделия и дополнительные принадлежности могут быть изменены в любое время с целью внесения улучшений и по другим причинам. Мы практикуем изменение номера модели в случае изменения ранее заявленных номинальных характеристик или свойств либо в случае существенного изменения конструкции. Тем не менее, некоторые технические характеристики изделия могут быть изменены без какого-либо уведомления. В спорном случае изделию может быть присвоен специальный номер модели для регистрации или определения особых характеристик, имеющих значение для конкретного случая применения изделия. Актуальные сведения о фактических технических характеристиках приобретаемого изделия можно получить у регионального представителя Omron.

Ошибки и опечатки

Информация, предоставляемая компанией Omron, предварительно проверяется и может считаться достоверной; тем не менее, компания Omron не несет ответственности за допущенные типографские и редакторские ошибки или опечатки.

Меры предосторожности и обеспечения безопасности

Сведения о мерах предосторожности и обеспечения безопасности см. в указанных ниже руководствах.

- *Серия NX, модули ЦПУ — Аппаратные средства. Руководство пользователя (Cat. No. W535)*
- *Серия NX, модуль ЦПУ NX102 — Аппаратные средства. Руководство пользователя (Cat. No. W593)*
- *Серия NX, модуль ЦПУ NX1P2 — Аппаратные средства. Руководство пользователя (Cat. No. W578)*
- *Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя (Cat No. W500)*

Меры предосторожности для обеспечения безопасной эксплуатации

Сведения о мерах предосторожности для обеспечения безопасной эксплуатации см. в указанных ниже руководствах.

- *Серия NX, модули ЦПУ — Аппаратные средства. Руководство пользователя (Cat. No. W535)*
- *Серия NX, модуль ЦПУ NX102 — Аппаратные средства. Руководство пользователя (Cat. No. W593)*
- *Серия NX, модуль ЦПУ NX1P2 — Аппаратные средства. Руководство пользователя (Cat. No. W578)*
- *Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя (Cat No. W500)*

Меры предосторожности для обеспечения надлежащей эксплуатации

Сведения о мерах предосторожности для обеспечения надлежащей эксплуатации см. в указанных ниже руководствах.

- *Серия NX, модули ЦПУ — Аппаратные средства. Руководство пользователя (Cat. No. W535)*
- *Серия NX, модуль ЦПУ NX102 — Аппаратные средства. Руководство пользователя (Cat. No. W593)*
- *Серия NX, модуль ЦПУ NX1P2 — Аппаратные средства. Руководство пользователя (Cat. No. W578)*
- *Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя (Cat No. W500)*

Директивы и стандарты

Сведения о соблюдении Директив и стандартов см. в указанных ниже руководствах.

- *Серия NX, модули ЦПУ — Аппаратные средства. Руководство пользователя (Cat. No. W535)*
- *Серия NX, модуль ЦПУ NX102 — Аппаратные средства. Руководство пользователя (Cat. No. W593)*
- *Серия NX, модуль ЦПУ NX1P2 — Аппаратные средства. Руководство пользователя (Cat. No. W578)*
- *Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя (Cat No. W500)*

Версии

Для упорядочения версий аппаратной части и программного обеспечения модулей Серия NJ/NX и ведомых устройств EtherCAT используются понятия «аппаратная версия» и «версия модуля». Аппаратная версия или версия модуля изменяются при внесении любого изменения в характеристики аппаратной или программной части. Даже если два модуля или два ведомых устройства EtherCAT будут иметь один и тот же номер модели, они не будут идентичны по функциональным или эксплуатационным характеристикам, если у них будут разные номера аппаратной версии или версии модуля.

Проверка версий

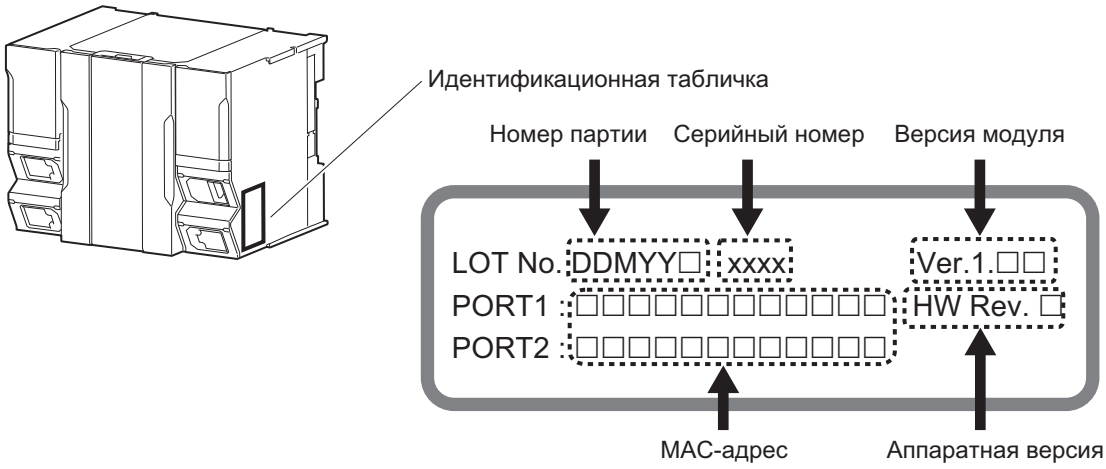
Номер версии можно посмотреть на идентификационной этикетке на изделии или узнать с помощью Sysmac Studio.

Проверка версии модуля с помощью идентификационной информации

Версия модуля указывается на этикетке с идентификационной информацией на боковой стенке изделия.

● Для NX701

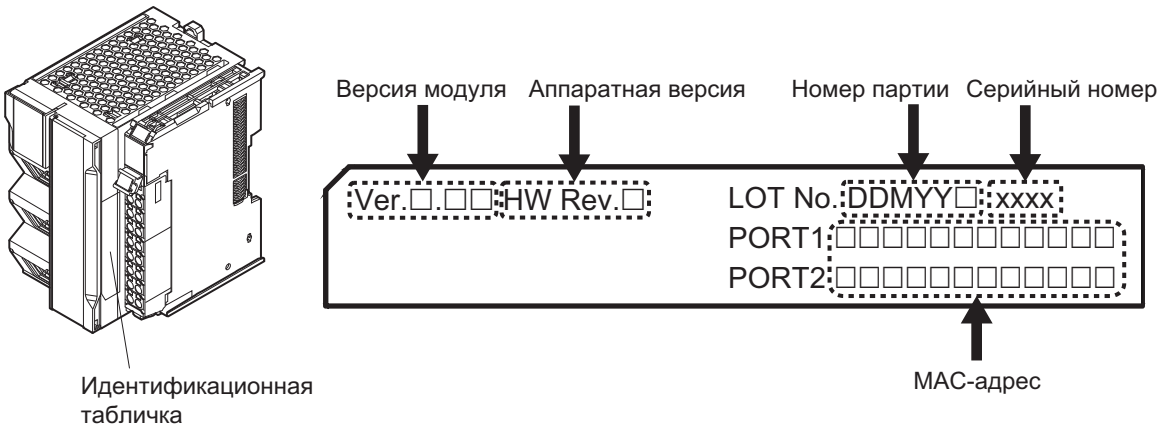
На рисунке ниже показан пример идентификационной информации для модуля ЦПУ NX701-□□00 серии NX.



Примечание Аппаратная версия модуля может не указываться, в этом случае поле аппаратной версии не заполняется.

● Для NX102

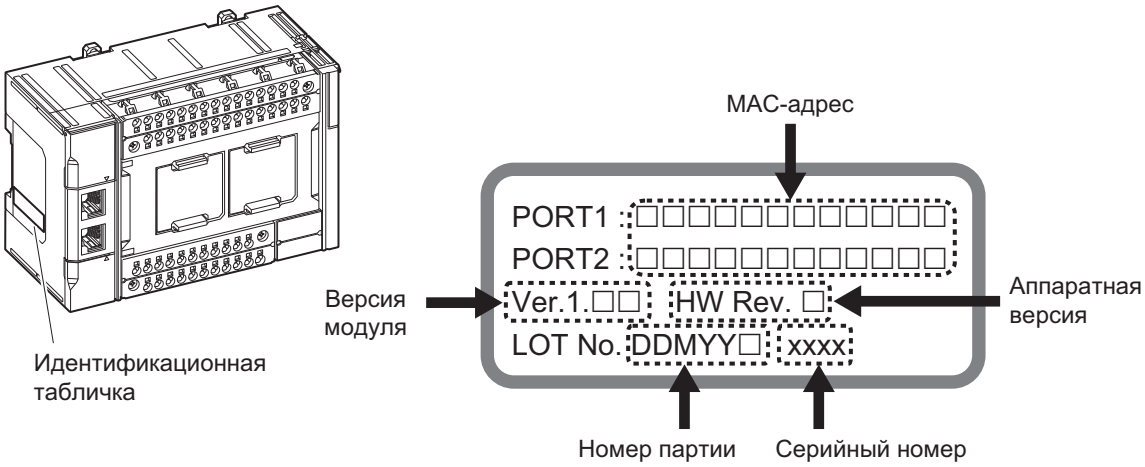
На рисунке ниже показан пример идентификационной информации для модуля ЦПУ NX102-□□00 серии NX.



Примечание Аппаратная версия модуля может не указываться, в этом случае поле аппаратной версии не заполняется.

● Для NX1P2

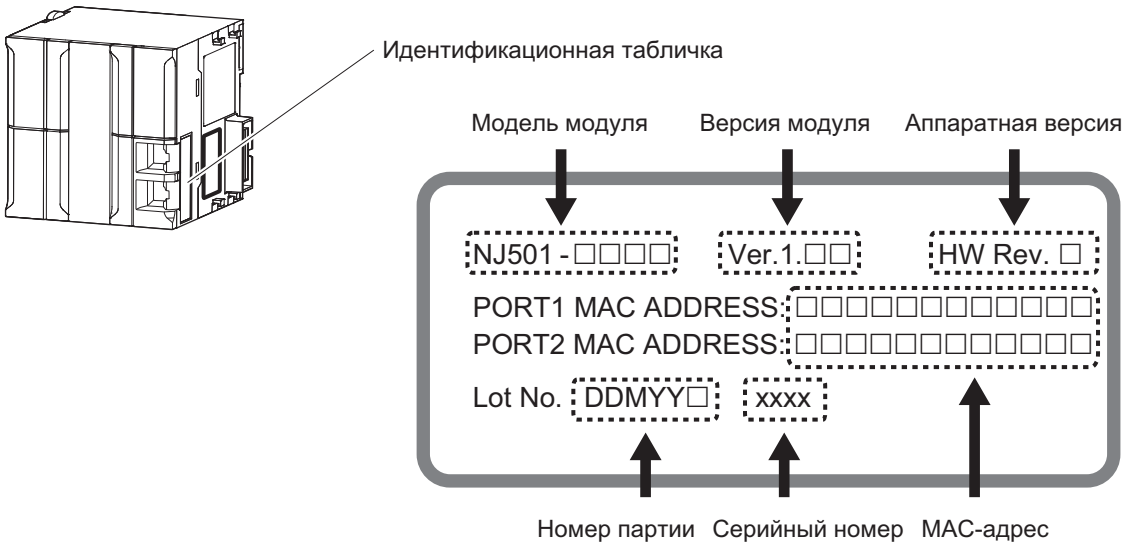
На рисунке ниже показан пример идентификационной информации для модуля ЦПУ NX1P2-□□□□□□□ серии NX.



Примечание Аппаратная версия модуля может не указываться, в этом случае поле аппаратной версии будет пустым.

● Для серии NJ

На рисунке ниже показан пример идентификационной информации для модуля ЦПУ NJ501-1□00 серии NJ.



Примечание Аппаратная версия модуля может не указываться, в этом случае поле аппаратной версии будет пустым.

Проверка версии модуля с помощью Sysmac Studio

Для проверки версии модуля можно использовать Sysmac Studio. Порядок действий для модулей отличается от порядка действий для ведомых устройств EtherCAT.

● Проверка версии модуля для модуля ЦПУ серии NX

Версию модуля для модуля можно посмотреть в окне производственной информации (Production Information) для этого модуля, установив связь с программным обеспечением Sysmac Studio. Это можно сделать для указанных ниже модулей.

Модель	Модуль, для которого можно проверить версию модуля
NX701-□□□□	Модуль ЦПУ
NX102-□□□□	Модуль ЦПУ и модуль NX в стойке ЦПУ
NX1P2-□□□□	Модуль ЦПУ, модуль NX в стойке ЦПУ и дополнительные платы

1 Щелкните правой кнопкой мыши **стойку ЦПУ (CPU Rack)** в разделе **Configurations and Setup (Конфигурации и настройка)** — **CPU/Expansion Racks (Стойки ЦПУ/стойки расширения)** в окне Multiview Explorer и выберите пункт **Production Information (Производственная информация)**.
Отобразится диалоговое окно Production Information (Производственная информация).

● Проверка версии модуля для модуля ЦПУ серии NJ

Версию модуля для модуля можно посмотреть в окне производственной информации (Production Information) для этого модуля, установив связь с программным обеспечением Sysmac Studio. Это можно сделать для модуля ЦПУ, специальных модулей ввода-вывода серии CJ и для модулей шины ЦПУ серии CJ. С помощью Sysmac Studio невозможно проверить версию модуля для базовых модулей ввода-вывода серии CJ.

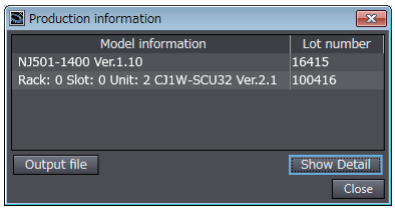
1 Дважды нажмите **стойку ЦПУ (CPU Rack)** в разделе **Configurations and Setup (Конфигурации и настройка)** — **CPU/Expansion Racks (Стойки ЦПУ/стойки расширения)** в окне Multiview Explorer. Либо щелкните правой кнопкой мыши **стойку ЦПУ (CPU Rack)** в разделе **Configurations and Setup (Конфигурации и настройка)** —

CPU/Expansion Racks (Стойки ЦПУ/стойки расширения) в окне Multiview Explorer и выберите пункт **Edit (Изменить)** в контекстном меню.
Отобразится окно редактора модулей (Unit Editor).

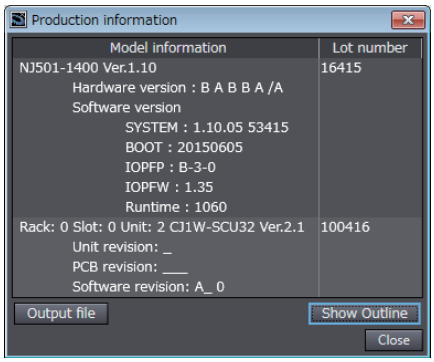
2 Щелкните правой кнопкой мыши в любом свободном месте в окне редактора модулей (Unit Editor) и выберите пункт **Production Information (Производственная информация)**.
Отобразится диалоговое окно Production Information (Производственная информация).

● Изменение сведений, отображаемых в диалоговом окне «Производственная информация»

1 Нажмите кнопку **Show Detail (Показать подробные сведения)** или **Show Outline (Показать общие сведения)** в правом нижнем углу диалогового окна Production Information (Производственная информация).
В зависимости от нажатой кнопки будет отображаться подробная или общая информация.



Общая информация



Подробная информация

Содержание общей и подробной информации отличается. В окне подробных сведений отображаются данные о версии модуля, версии оборудования и о различных версиях. В окне общей информации отображается только версия модуля.

Примечание Аппаратная версия отображается справа от версии оборудования и отделяется от нее наклонной чертой («/»). Аппаратная версия модуля может не указываться, в этом случае поле аппаратной версии будет пустым.

● Проверка версии модуля для ведомого устройства EtherCAT

Версию модуля для ведомого устройства EtherCAT можно посмотреть в производственной информации (Production Information) для этого устройства, установив связь с программным обеспечением Sysmac Studio.
Для просмотра версии модуля используйте приведенный ниже порядок действий.

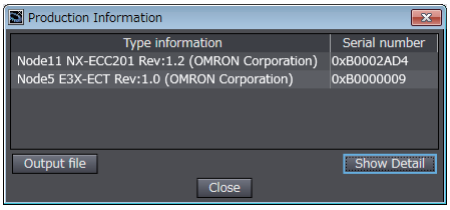
1 Дважды щелкните **EtherCAT** в разделе **Configurations and Setup (Конфигурации и настройка)** в окне Multiview Explorer. Либо щелкните правой кнопкой мыши пункт **EtherCAT** в разделе **Configurations and Setup (Конфигурации и настройка)** и выберите пункт **Edit (Изменить)** в контекстном меню.
Отобразится вкладка EtherCAT.

2 Щелкните правой кнопкой мыши ведущее устройство на вкладке EtherCAT и выберите пункт **Display Production Information (Отобразить производственные данные)**.

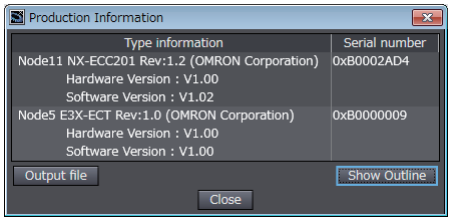
Отобразится диалоговое окно Production Information (Производственная информация). Версия модуля отображается после слова «Rev.»

● Изменение сведений, отображаемых в диалоговом окне «Производственная информация»

1 Нажмите кнопку **Show Detail (Показать подробные сведения)** или **Show Outline (Показать общие сведения)** в правом нижнем углу диалогового окна Production Information (Производственная информация).
В зависимости от нажатой кнопки будет отображаться подробная или общая информация.



Общая информация



Подробная информация

Взаимосвязь между версиями модулей ЦПУ и версиями Sysmac Studio

Состав поддерживаемых функций зависит от версии модуля Серия NJ/NX, модули ЦПУ. Для использования функций, добавленных при обновлении версии, также требуется версия Sysmac Studio, которая поддерживает эти функции.
Информацию о взаимосвязи между версиями модулей Модуль ЦПУ и версиями Sysmac Studio, а также сведения о функциях, которые поддерживаются в каждой версии модуля, см. в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Сопутствующие руководства

Ниже приводится список руководств, имеющих отношение к настоящему руководству. Используйте эти руководства для поиска необходимой информации. Используйте эти руководства для поиска необходимой информации.

Название руководства	Cat. No.	Номера моделей	Применение	Описание
Серия NX, модули ЦПУ — Аппаратные средства. Руководство пользователя	W535	NX701-□□□□	Ознакомление с основными характеристиками модулей ЦПУ NX701, получение общей информации о проектировании, монтаже и обслуживании. Предоставляется информация преимущественно об аппаратном обеспечении.	Предоставляется вводная информация о системе NX701 в целом, включая следующую информацию о модуле ЦПУ: • свойства и конфигурация системы; • вводная информация; • названия и функции элементов; • общие характеристики; • механический и электрический монтаж; • обслуживание и проверка.
Серия NX, модуль ЦПУ NX102 — Аппаратные средства. Руководство пользователя	W593	NX102-□□□□	Ознакомление с основными характеристиками модулей ЦПУ NX102, получение общей информации о проектировании, монтаже и обслуживании. Предоставляется информация преимущественно об аппаратном обеспечении.	Предоставляется вводная информация о системе NX102 в целом, включая следующую информацию о модуле ЦПУ: • свойства и конфигурация системы; • вводная информация; • названия и функции элементов; • общие характеристики; • механический и электрический монтаж; • обслуживание и проверка.
Серия NX, модуль ЦПУ NX1P2 — Аппаратные средства. Руководство пользователя	W578	NX1P2-□□□□	Ознакомление с основными характеристиками модулей ЦПУ NX1P2, получение общей информации о проектировании, монтаже и обслуживании. Предоставляется информация преимущественно об аппаратном обеспечении.	Предоставляется вводная информация о системе NX1P2 в целом, включая следующую информацию о модуле ЦПУ: • свойства и конфигурация системы; • вводная информация; • названия и функции элементов; • общие характеристики; • механический и электрический монтаж; • обслуживание и проверка.
Серия NJ, модули ЦПУ — Аппаратные средства. Руководство пользователя	W500	NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Ознакомление с основными характеристиками модулей ЦПУ серии NJ, получение общей информации о проектировании, монтаже и обслуживании. Предоставляется информация преимущественно об аппаратном обеспечении.	Предоставляется вводная информация о системе серии NJ в целом, включая следующую информацию о модуле ЦПУ: • свойства и конфигурация системы; • вводная информация; • названия и функции элементов; • общие характеристики; • механический и электрический монтаж; • обслуживание и проверка.

Название руководства	Cat. No.	Номера моделей	Применение	Описание
Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя	W501	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Получение информации о программировании и настройке модуля ЦПУ серии NJ/NX. Предоставляется информация преимущественно о программном обеспечении.	Предоставляется следующая информация о контроллере на базе модуля ЦПУ серии NJ/NX: • работа модуля ЦПУ; • свойства и функции модуля ЦПУ; • начальная настройка параметров; • программирование на языках стандарта IEC 61131-3.
Серия NX, модуль ЦПУ NX1P2 — Встроенные входы-выходы и дополнительная плата. Руководство пользователя	W579	NX1P2-□□□□	Подробное изучение функций только модуля ЦПУ NX1P2 серии NX и получение общих сведений о функциях модулей ЦПУ серии NJ/NX.	Предоставляется следующая информация о функциях модуля ЦПУ NX1P2: • встроенные входы-выходы; • дополнительные платы последовательного интерфейса; • дополнительные платы аналоговых входов-выходов. Также предлагается вводная информация о следующих функциях модулей ЦПУ серии NJ/NX: • функции управления движением; • функции для обмена данными по сети EtherNet/IP; • функции для обмена данными по сети EtherCAT.
Серия NJ/NX — Команды программирования. Справочное руководство	W502	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Подробное ознакомление с основными командами программирования модуля ЦПУ серии NJ/NX.	Описываются команды, входящие в набор команд (в соответствии со стандартом IEC 61131-3).
Серия NJ/NX, модули ЦПУ — Управление движением. Руководство пользователя	W507	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Получение информации о настраиваемых параметрах и принципах программирования системы управления движением.	Описываются параметры и работа модуля ЦПУ, а также принципы создания программы для управления движением.
Серия NJ/NX — Команды управления движением. Справочное руководство	W508	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Изучение команд управления движением.	Описываются команды программирования, предназначенные для управления движением.
Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT®. Руководство пользователя	W505	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Применение встроенного порта EtherCAT модуля ЦПУ серии NJ/NX.	Предоставляются сведения о встроенном порте EtherCAT. Данное руководство содержит вводные сведения, а также сведения о конфигурации, функциях и настройке.
Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP™. Руководство пользователя	W506	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Применение встроенного порта EtherNet/IP модуля ЦПУ серии NJ/NX.	Предоставляются сведения о встроенном порте EtherNet/IP. Предоставляются сведения о настройке основных параметров, обмене данными через логические связи тегов и о других функциях.
Серия NJ/NX, модули ЦПУ — OPC UA. Руководство пользователя	W588	NX701-□□□□ NX102-□□□□ NJ501-1□00	Использование OPC UA.	Приводится описание OPC UA.
Серия NX, модули ЦПУ — Протокол FINS. Руководство пользователя	W596	NX701-□□20 NX102-□□□□	Использование функции FINS модуля ЦПУ серии NX.	Описывается функция FINS модуля ЦПУ серии NX.

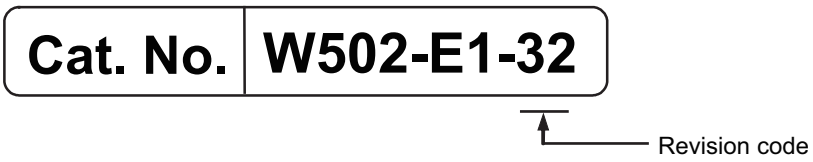
Название руководства	Cat. No.	Номера моделей	Применение	Описание
Серия NJ/NX, модули ЦПУ — Подключение к базе данных. Руководство пользователя	W527	NX701-□□20 NX102-□□20 NJ501-□□20 NJ101-□□20	Использование службы подключения к базе данных с контроллерами серии NJ/NX.	Содержит описание службы подключения к базе данных.
Серия NJ, модули ЦПУ — SECS/GEM. Руководство пользователя	W528	NJ501-1340	Использование служб GEM с контроллерами серии NJ.	Предоставляются сведения о службах GEM.
Серия NJ, модули ЦПУ — Встроенные функции управления роботами. Руководство пользователя	O037	NJ501-R□□□	Применение модуля ЦПУ серии NJ со встроенными функциями для управления роботами.	Описываются параметры и работа модуля ЦПУ, а также принципы создания программ для управления роботами OMRON.
Sysmac Studio — Функции для построения интегрированных роботизированных систем автоматизации с использованием модуля ЦПУ со встроенными функциями управления роботами — Руководство по работе	W595	SYSMAC-SE2□□□ SYSMAC-SE200D-64	Изучение порядка работы с программным обеспечением Sysmac Studio и его функций, предназначенных для создания и настройки интегрированной роботизированной системы автоматизации с применением модуля ЦПУ со встроенными функциями для управления роботами.	Описывается порядок работы с ПО Sysmac Studio применительно к модулю ЦПУ со встроенными функциями для управления роботами.
Sysmac Studio — Функции для построения интегрированных роботизированных систем автоматизации с использованием прикладного контроллера на базе промышленного ПК. Руководство по работе	W621	SYSMAC-SE2□□□ SYSMAC-SE200D-64	Изучение порядка работы с программным обеспечением Sysmac Studio и его функций, предназначенных для создания и настройки интегрированной роботизированной системы автоматизации с применением прикладного контроллера на базе промышленного ПК.	Описывается порядок работы с ПО Sysmac Studio применительно к прикладному контроллеру на базе промышленного ПК.
Sysmac Studio — Функция 3D-моделирования. Руководство по работе	W618	SYSMAC-SE2□□□ SYSMAC-SA4□□□-64	Ознакомление с предусмотренной в Sysmac Studio функцией трехмерного моделирования и принципами ее использования.	Приводится общее описание функции трехмерного моделирования, предусмотренной в Sysmac Studio, описываются принципы ее работы и порядок работы с ней.
Серия NJ, модули ЦПУ — NJ Robotics. Руководство пользователя	W539	NJ501-4□□□ NJ501-R□□□	Управление роботами с использованием модулей ЦПУ серии NJ.	Описываются функции для управления роботами.
Серия NJ/NY, интегрированный контроллер ЧПУ — Руководство пользователя	O030	NJ501-5300 NY532-5400	Реализация числового программного управления с использованием контроллеров серии NJ/NY.	Описываются функции для осуществления числового программного управления.
Серия NJ/NY — Команды языка «G-код». Справочное руководство	O031	NJ501-5300 NY532-5400	Изучение спецификаций команд языков программирования «G-код»/«M-код».	Описываются команды языков программирования «G-код»/«M-код».

Название руководства	Cat. No.	Номера моделей	Применение	Описание
Серия NJ/NX — Поиск и устранение неполадок. Руководство	W503	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Получение информации об ошибках, которые могут возникать в контроллере серии NJ/NX.	Описываются принципы диагностики и устранения ошибок, которые могут возникать в работе контроллера серии NJ/NX; приводятся сведения об отдельных ошибках.
Sysmac Studio, версия 1 — Руководство по работе	W504	SYSMAC -SE2□□□	Изучение порядка работы с программным обеспечением Sysmac Studio и его функций.	Описывается порядок работы с ПО Sysmac Studio.
CNC Operator — Руководство по работе	O032	SYSMAC-RTNC0□ □□D	Вводная информация о приложении CNC Operator и о том, как его использовать.	Вводная информация о приложении CNC Operator; описание порядка установки, основных операций, операций по подключению и рабочих процедур, связанных с основными функциями приложения.
Серия NX, интерфейсный модуль EtherCAT® — Руководство пользователя	W519	NX-ECC□□□	Изучение принципов работы с интерфейсным модулем EtherCAT серии NX и ведомыми терминалами EtherCAT.	Приводится следующая информация: общее описание системы и методов настройки ведомого терминала EtherCAT (который состоит из интерфейсного модуля EtherCAT серии NX и модулей NX); сведения об оборудовании и настройке; сведения о функциях для настройки, мониторинга и управления модулями NX по сети EtherCAT.
Серия NX — Справочное руководство	W525	NX-□□□□□□	Справочные данные, необходимые для настройки систем с модулями серии NX.	Предоставляются данные о потребляемой мощности, массах и другие технические данные модулей NX, которые требуются для проектирования и настройки систем с модулями серии NX.
Серия NX, модули NX — Руководство пользователя	W521	NX-ID□□□□ NX-IA□□□□ NX-OC□□□□ NX-OD□□□□ NX-MD□□□□	Получение информации об использовании модулей NX.	Описание аппаратной части, способов настройки и функций модулей NX. Руководства доступны для следующих модулей. Модули дискретных входов-выходов, модули аналоговых входов-выходов, конструктивные модули, модули интерфейса позиционирования, модули интерфейсов связи, модули входов тензодатчика и модули ведущего устройства IO-Link.
	W522	NX-AD□□□□ NX-DA□□□□		
	W592	NX-HAD□□□		
	W566	NX-TS□□□□ NX-HB□□□□		
	W523	NX-PD1□□□ NX-PF0□□□ NX-PC0□□□ NX-TBX01		
	W524	NX-EC0□□□ NX-ECS□□□ NX-PG0□□□		
	W540	NX-CIF□□□		
	W565	NX-RS□□□□		
	W567	NX-ILM□□□		

Название руководства	Cat. No.	Номера моделей	Применение	Описание
Серия CJ, специальный модуль Руководства для модуля ЦПУ серии NJ	W490	CJ1W-AD□□□ CJ1W-DA□□□ CJ1W-MAD42	Получение информации об использовании модулей серии CJ с модулями ЦПУ серии NJ.	Описываются способы использования модулей серии CJ с модулем ЦПУ серии NJ, включая способы доступа и интерфейсы программирования, а также связанные с этим необходимые меры предосторожности. Руководства доступны для следующих модулей. Модули аналоговых входов-выходов, модули аналоговых входов-выходов с гальванической развязкой, модули регулирования температуры, модули датчиков идентификации, модули скоростных счетчиков, модули последовательного интерфейса, модули DeviceNet, модули EtherNet/IP и модули ведущего устройства CompoNet.
	W491	CJ1W-TC□□□		
	W492	CJ1W-CT021		
	W498	CJ1W-PDC15 CJ1W-PH41U CJ1W-AD04U		
	W493	CJ1W-CRM21		
	W494	CJ1W-SCU□□		
	W495	CJ1W-EIP21		
	W497	CJ1W-DRM21		
	Z317	CJ1W-V680□□□		
CX-Protocol — Руководство по работе	W344	---	Создание протоколов передачи данных для устройств общего назначения, подключенных к модулям последовательного интерфейса серии CJ.	Описываются способы и порядок работы с программой CX-Protocol.
Серия GX, модули ведомого устройства EtherCAT — Руководство пользователя	W488	GX-ID□□□□ GX-OD□□□□ GX-OC□□□□ GX-MD□□□□ GX-AD□□□□ GX-DA□□□□ GX-EC□□□□ XWT-ID□□ XWT-OD□□	Изучение порядка эксплуатации терминалов удаленного ввода-вывода EtherCAT.	Описываются аппаратные средства, способы настройки и функции терминалов удаленного ввода-вывода EtherCAT.
Серия 1S, серводвигатели и сервоприводы переменного тока со встроенным портом EtherCAT® — Связь. Руководство пользователя	I586	R88M-1□ R88D-1SN□-ECT	Изучение принципов работы с серводвигателями и сервоприводами со встроенным интерфейсом связи EtherCAT.	Описываются аппаратные средства, способы настройки и функции серводвигателей и сервоприводов со встроенным интерфейсом связи EtherCAT.
	I621	R88M-1AL□/ -1AM□ R88D-1SAN□-ECT		
Серия G5, серводвигатели и сервоприводы переменного тока со встроенным портом EtherCAT® — Связь. Руководство пользователя	I576	R88M-K□ R88D-KN□-ECT	Изучение принципов работы с серводвигателями и сервоприводами переменного тока со встроенным интерфейсом связи EtherCAT.	Описываются аппаратные средства, способы настройки и функции серводвигателей и сервоприводов переменного тока со встроенным интерфейсом связи EtherCAT. Серия G5 включает модели двигателей линейного типа и специализированные модели для позиционирования.
	I577	R88L-EC-□ R88D-KN□-ECT-L		

Перечень версий

Версия руководства указывается в конце номера каталога на передней и задней сторонах обложки руководства.



Обозначение версии	Дата	Изменения
01	Июль, 2011 г.	Оригинальная версия
02	Сентябрь, 2011 г.	Исправлены ошибки.
03	Март, 2012 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.01 и ПО Sysmac Studio версии 1.02. • Исправлены ошибки.
04	Май, 2012 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.02 и ПО Sysmac Studio версии 1.03. • Исправлены ошибки.
05	Август, 2012 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.03 и ПО Sysmac Studio версии 1.04. • Исправлены ошибки.
06	Февраль, 2013 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.04 и ПО Sysmac Studio версии 1.05. • Исправлены ошибки.
07	Апрель, 2013 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.05 и ПО Sysmac Studio версии 1.06. • Исправлены ошибки.
08	Июнь, 2013 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.06 и ПО Sysmac Studio версии 1.07. • Исправлены ошибки.
09	Сентябрь, 2013 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.07 и ПО Sysmac Studio версии 1.08. • Исправлены ошибки.
10	Декабрь, 2013 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.08 и ПО Sysmac Studio версии 1.09. • Исправлены ошибки.
11	Июль, 2014 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.09 и ПО Sysmac Studio версии 1.10. • Исправлены ошибки.
12	Январь, 2015 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.10 и ПО Sysmac Studio версии 1.12. • Исправлены ошибки.
13	Апрель, 2015 г.	• Внесены изменения в связи с добавлением модулей ЦПУ NX701-□□□□ серии NX и выпуском ПО Sysmac Studio версии 1.13. • Добавлена информация о модулях ЦПУ NJ101-□□□□ серии NJ. • Исправлены ошибки.
14	Октябрь, 2015 г.	• Внесены изменения в связи с добавлением аппаратной версии. • Исправлены ошибки.

Обозначение версии	Дата	Изменения
15	Апрель, 2016 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.11 и ПО Sysmac Studio версии 1.15. • Исправлены ошибки.
16	Июль, 2016 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.12 и ПО Sysmac Studio версии 1.16. • Исправлены ошибки.
17	Октябрь, 2016 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.13 и ПО Sysmac Studio версии 1.17. • Исправлены ошибки.
18	Ноябрь, 2016 г.	Исправлены ошибки.
19	Январь, 2017 г.	Исправлены ошибки.
20	Апрель, 2017 г.	Исправлены ошибки.
21	Октябрь, 2017 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.16. • Исправлены ошибки.
22	Апрель, 2018 г.	• Внесены изменения в связи с выпуском модуля ЦПУ с версией модуля 1.18 и ПО Sysmac Studio версии 1.22. • Исправлены ошибки.
23	Апрель, 2018 г.	• Внесены изменения в связи с добавлением модулей ЦПУ NX102 серии NX. • Внесены изменения в связи с переносом информации о кодах событий в *Серия NJ/NX — Поиск и устранение неполадок. Руководство (Cat. No. W503)*. • Исправлены ошибки.
24	Июль, 2018 г.	Исправлены ошибки.
25	Январь, 2019 г.	Исправлены ошибки.
26	Апрель, 2019 г.	• Добавлена информация о функциях, поддерживаемых модулями ЦПУ NX102 с версией модуля 1.32 или более поздней версией. • Добавлена информация о функциях, поддерживаемых модулями ЦПУ NX1P2 с версией модуля 1.21 или более поздней версией. • Добавлена информация о функциях, поддерживаемых модулями ЦПУ NJ501-1□00, NJ301-□□□□ и NJ101-□□00 с версией модуля 1.21 или более поздней версией.
27	Июль, 2019 г.	• Внесены изменения в связи с обновлением версии модуля модулей ЦПУ NX102-□□00, NX1P2-□□□□□□, NJ501-1□00, NJ301-□□□□ и NJ101-□□00 до версии 1.40. • Внесены изменения в связи с обновлением версии модуля модулей ЦПУ NX701-□□□□, NJ501-4□00, NJ501-4□10, NJ501-1340 и NJ501-5300 до версии 1.21.
28	Июль, 2019 г.	Исправлены ошибки.
29	Октябрь, 2019 г.	Исправлены ошибки.
30	Август, 2020 г.	• Внесены изменения в связи с добавлением модулей ЦПУ NJ501-R□□□. • Исправлены ошибки.
31	Октябрь, 2020 г.	Исправлены ошибки.
32	Январь, 2021 г.	Исправлены ошибки.

1

Набор команд

В данном разделе в табличном виде приводится краткое описание команд, которые могут использоваться со следующим устройством: Серия NJ/NX, модули ЦПУ.

Набор команд

- Описание команд управления движением см. в документе *Серия NJ/NX — Команды программирования для управления движением. Справочное руководство (Cat. No. W508)*.
- Описание команд моделирования см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Команды языка релейно-контактных схем

Команда	Имя	Функция	Стр.
LD	Загрузка	Считывает значение переменной типа BOOL.	стр. 2-16
LDN	Загрузка НЕ	Считывает инвертированное значение переменной типа BOOL.	стр. 2-16
AND	И	Выполняет операцию «логическое И» над значением переменной типа BOOL и входным значением.	стр. 2-19
ANDN	И НЕ	Выполняет операцию «логическое И» над инвертированным значением переменной типа BOOL и входным значением.	стр. 2-19
OR	ИЛИ	Выполняет операцию «логическое ИЛИ» над значением переменной типа BOOL и условием выполнения.	стр. 2-22
ORN	ИЛИ НЕ	Выполняет операцию «логическое ИЛИ» над инвертированным значением переменной типа BOOL и условием выполнения.	стр. 2-22
Out	Выход	Берет логический результат выполнения предыдущей команды и выводит его в переменную типа BOOL.	стр. 2-25
OutNot	Выход НЕ	Берет инвертированное значение логического результата выполнения предыдущей команды и выводит его в переменную типа BOOL.	стр. 2-25

Команды для инструкций языка ST

Команда	Имя	Функция	Стр.
IF	Если	Выбирает для выполнения одну из двух инструкций на основе результата оценки указанного выражения условия.	стр. 2-30
CASE	Case	Выбирает для выполнения одну из нескольких инструкций в зависимости от значения указанного целочисленного выражения.	стр. 2-34
WHILE	Цикл While	Повторно выполняет некоторую инструкцию, пока результат оценки указанного выражения условия равен ИСТИНА.	стр. 2-38
REPEAT	Цикл Repeat	Выполняет некоторую инструкцию один раз, а затем выполняет ее повторно до тех пор, пока указанное выражение условия не становится равным ИСТИНА.	стр. 2-41
EXIT	Выход из цикла	Завершает выполнение самого внутреннего цикла, созданного командой циклической обработки FOR, WHILE или REPEAT.	стр. 2-44
RETURN	Возврат	Завершает функцию или функциональный блок и возвращает программу к вызвавшей их команде.	стр. 2-47
FOR	Начало цикла	Указывает условие для повторного выполнения и повторно выполняет инструкции, заключенные между FOR и END_FOR.	стр. 2-48

Входные битовые команды

Команда	Имя	Функция	Стр.
R_TRIG (Up)	Триггер по полож. фронту	Выдает состояние ИСТИНА только в течение одного цикла выполнения задачи, когда состояние входного сигнала меняется на ИСТИНА.	стр. 2-50
F_TRIG (Down)	Триггер по отриц. фронту	Выдает состояние ИСТИНА только в течение одного цикла выполнения задачи, когда состояние входного сигнала меняется на ЛОЖЬ.	стр. 2-50
TestABit	Проверка бита A	Выводит значение указанного бита в битовой строке.	стр. 2-54
TestABitN	Проверка бита A НЕ	Выводит инвертированное значение указанного бита в битовой строке.	стр. 2-54

Выходные битовые команды

Команда	Имя	Функция	Стр.
RS	Удержание с приоритетом сброса	Хранит значение переменной типа BOOL. Если состояние ИСТИНА одновременно присутствует на входе Set и входе Reset, приоритетом обладает вход Reset.	стр. 2-58
SR	Удержание с приоритетом установки	Хранит значение переменной типа BOOL. Если состояние ИСТИНА одновременно присутствует на входе Set и входе Reset, приоритетом обладает вход Set.	стр. 2-61
Set	Установка	Переводит переменную типа BOOL в состояние ИСТИНА.	стр. 2-64
Reset	Сброс	Переводит переменную типа BOOL в состояние ЛОЖЬ.	стр. 2-64
SetBits	Установка битов	Переводит группу смежных битов в битовой строке в состояние ИСТИНА.	стр. 2-68
ResetBits	Сброс битов	Переводит группу смежных битов в битовой строке в состояние ЛОЖЬ.	стр. 2-68
SetABit	Установка бита	Переводит указанный бит в битовой строке в состояние ИСТИНА.	стр. 2-71
ResetABit	Сброс бита	Переводит указанный бит в битовой строке в состояние ЛОЖЬ.	стр. 2-71
OutABit	Вывод бита	Переводит указанный бит в битовой строке в состояние ИСТИНА или ЛОЖЬ.	стр. 2-74

Команды управления последовательностью выполнения

Команда	Имя	Функция	Стр.
End	Конец	Завершает выполнение программы в текущем цикле выполнения задачи.	стр. 2-78
RETURN	Возврат	Завершает функцию или функциональный блок и возвращает программу к вызвавшей их команде.	стр. 2-79
MC	Начало главного управления	Обозначает точку начала области главного управления и сбрасывает область главного управления.	стр. 2-80
MCR	Конец главного управления	Обозначает точку конца области главного управления.	стр. 2-80
JMP	Переход	Производит переход в указанную точку программы.	стр. 2-94
FOR	Начало цикла	Обозначает начальную точку циклического выполнения и указывает условие повтора.	стр. 2-96
NEXT	Конец цикла	Обозначает конечную точку циклического выполнения.	стр. 2-96

Команда	Имя	Функция	Стр.
BREAK	Выход из цикла	Прекращает работу цикла, созданного самой внутренней командой FOR и следующей по порядку командой NEXT.	стр. 2-104

Команды сравнения

Команда	Имя	Функция	Стр.
EQ (=)	Равно	Определяет, равны ли значения двух или большего числа переменных.	стр. 2-108
NE (<>)	Не равно	Определяет неравенство значений двух переменных.	стр. 2-111
LT (<)	Меньше	Выполняет сравнение «меньше» для двух значений.	стр. 2-114
LE (<=)	Меньше или равно	Выполняет сравнение «меньше или равно» для двух значений.	стр. 2-114
GT (>)	Больше	Выполняет сравнение «больше» для двух значений.	стр. 2-114
GE (>=)	Больше или равно	Выполняет сравнение «больше или равно» для двух значений.	стр. 2-114
EQascii	Сравнение текстовых строк: равно	Определяет, совпадают ли друг с другом две или более текстовых строк.	стр. 2-118
NEascii	Сравнение текстовых строк: не равно	Определяет, отличаются ли друг от друга две текстовые строки.	стр. 2-120
LTascii	Сравнение текстовых строк: меньше	Выполняет сравнение «меньше» для двух текстовых строк.	стр. 2-122
LEascii	Сравнение текстовых строк: меньше или равно	Выполняет сравнение «меньше или равно» для двух текстовых строк.	стр. 2-122
GTascii	Сравнение текстовых строк: больше	Выполняет сравнение «больше» для двух текстовых строк.	стр. 2-122
GEascii	Сравнение текстовых строк: больше или равно	Выполняет сравнение «больше или равно» для двух текстовых строк.	стр. 2-122
Cmp	Сравнение	Сравнивает два значения.	стр. 2-125
ZoneCmp	Попадание в диапазон	Определяет, находится ли сравниваемое значение между указанными максимальным и минимальным значениями.	стр. 2-128
TableCmp	Таблица сравнения	Проверяет, принадлежит ли заданное значение тому или иному диапазону, которые заданы в таблице сравнения.	стр. 2-131
AryCmpEQ	Сравнение массивов: равно	Определяет, равны ли друг другу соответствующие элементы двух массивов.	стр. 2-134
AryCmpNE	Сравнение массивов: не равно	Определяет, отличаются ли друг от друга соответствующие элементы двух массивов.	стр. 2-134
AryCmpLT	Сравнение массивов: меньше	Выполняет сравнение «меньше» для соответствующих элементов двух массивов.	стр. 2-137
AryCmpLE	Сравнение массивов: меньше или равно	Выполняет сравнение «меньше или равно» для соответствующих элементов двух массивов.	стр. 2-137
AryCmpGT	Сравнение массивов: больше	Выполняет сравнение «больше» для соответствующих элементов двух массивов.	стр. 2-137
AryCmpGE	Сравнение массивов: больше или равно	Выполняет сравнение «больше или равно» для соответствующих элементов двух массивов.	стр. 2-137
AryCmpEQV	Сравнение массива со значением: равно	Определяет, равен ли каждый элемент массива заданному значению.	стр. 2-140

Команда	Имя	Функция	Стр.
AryCmpNEV	Сравнение массива со значением: не равно	Определяет, отличается ли каждый элемент массива от заданного значения.	стр. 2-140
AryCmpLTV	Сравнение массива со значением: меньше	Выполняет сравнение «меньше» для каждого элемента массива и заданного значения.	стр. 2-143
AryCmpLEV	Сравнение массива со значением: меньше или равно	Выполняет сравнение «меньше или равно» для каждого элемента массива и заданного значения.	стр. 2-143
AryCmpGTV	Сравнение массива со значением: больше	Выполняет сравнение «больше» для каждого элемента массива и заданного значения.	стр. 2-143
AryCmpGEV	Сравнение массива со значением: больше или равно	Выполняет сравнение «больше или равно» для каждого элемента массива и заданного значения.	стр. 2-143

Команды таймеров

Команда	Имя	Функция	Стр.
TON	Таймер задержки включения	Выдает состояние «ИСТИНА» по истечении заданного времени с момента запуска таймера.	стр. 2-148
TOF	Таймер задержки выключения	Выдает состояние «ЛОЖЬ» по истечении заданного времени с момента запуска таймера.	стр. 2-154
TP	Таймер импульса	Выдает состояние «ИСТИНА» в течение заданного времени с момента запуска таймера.	стр. 2-158
AccumulationTimer	Накопительный таймер	Измеряет общее время, в течение которого вход таймера находился в состоянии «ИСТИНА».	стр. 2-162
Timer	100 мс таймер	Выдает состояние «ИСТИНА» по истечении заданного времени с момента запуска таймера. Время задается с шагом 100 мс.	стр. 2-166

Команды счетчиков

Команда	Имя	Функция	Стр.
CTD	Обратный счетчик	Реализует счетчик обратного счета. Текущее значение счетчика уменьшается при поступлении сигнала на вход счетчика. Предустановленное и текущее значения счетчика должны относиться к типу данных INT.	стр. 2-170
CTD_**	Групповой обратный счетчик	Реализует счетчик обратного счета. Текущее значение счетчика уменьшается при поступлении сигнала на вход счетчика. Предустановленное и текущее значения счетчика должны относиться к одному из следующих типов данных: DINT, LINT, UDINT или ULINT.	стр. 2-173
CTU	Прямой счетчик	Реализует счетчик обратного счета. Текущее значение счетчика увеличивается при поступлении сигнала на вход счетчика. Предустановленное и текущее значения счетчика должны относиться к типу данных INT.	стр. 2-176
CTU_**	Групповой прямой счетчик	Реализует счетчик обратного счета. Текущее значение счетчика увеличивается при поступлении сигнала на вход счетчика. Предустановленное и текущее значения счетчика должны относиться к одному из следующих типов данных: DINT, LINT, UDINT или ULINT.	стр. 2-179

Команда	Имя	Функция	Стр.
CTUD	Прямой-обратный счетчик	Реализует счетчик, ведущий счет в прямом и обратном направлениях в соответствии с сигналами на входах прямого и обратного счета. Предустановленное и текущее значения счетчика должны относиться к типу данных INT.	стр. 2-182
CTUD_**	Групповой прямой-обратный счетчик	Реализует счетчик, ведущий счет в прямом и обратном направлениях в соответствии с сигналами на входах прямого и обратного счета. Предустановленное и текущее значения счетчика должны относиться к одному из следующих типов данных: DINT, LINT, UDINT или ULINT.	стр. 2-187

Команды математических операций

Команда	Имя	Функция	Стр.
ADD (+)	Сложение	Складывает целые или вещественные числа. Также объединяет текстовые строки.	стр. 2-195
AddOU (+OU)	Сложение с проверкой на переполнение	Складывает целые или вещественные числа. Также проверяет результат сложения целых чисел на переполнение.	стр. 2-200
SUB (-)	Вычитание	Вычитает целые или вещественные числа.	стр. 2-204
SubOU (-OU)	Вычитание с проверкой на переполнение	Вычитает целые или вещественные числа. Также проверяет результат вычитания целых чисел на переполнение.	стр. 2-208
MUL (*)	Умножение	Умножает целые или вещественные числа.	стр. 2-212
MulOU (*OU)	Умножение с проверкой на переполнение	Умножает целые и вещественные числа и выводит результат. Также проверяет результат умножения целых чисел на переполнение.	стр. 2-216
DIV (/)	Деление	Делит целые или вещественные числа.	стр. 2-220
MOD	Деление по модулю	Находит остаток от деления целых чисел.	стр. 2-224
ABS	Абсолютное значение	Находит абсолютное значение целого или вещественного числа.	стр. 2-226
RadToDeg	Радианы в градусы	Преобразует вещественное число из радианов (рад) в градусы (°).	стр. 2-228
DegToRad	Градусы в радианы	Преобразует вещественное число из градусов (°) в радианы (рад).	стр. 2-228
SIN	Синус в радианах	Вычисляет синус вещественного числа.	стр. 2-231
COS	Косинус в радианах	Вычисляет косинус вещественного числа.	стр. 2-231
TAN	Тангенс в радианах	Вычисляет тангенс вещественного числа.	стр. 2-231
ASIN	Арксинус (SIN^{-1})	Вычисляет арксинус (sin^{-1}) вещественного числа.	стр. 2-234
ACOS	Арккосинус (COS^{-1})	Вычисляет арккосинус (cos^{-1}) вещественного числа.	стр. 2-234
ATAN	Арктангенс (TAN^{-1})	Вычисляет арктангенс (tan^{-1}) вещественного числа.	стр. 2-234
SQRT	Квадратный корень	Вычисляет квадратный корень вещественного числа.	стр. 2-237
LN	Натуральный логарифм	Вычисляет натуральный логарифм вещественного числа.	стр. 2-240
LOG	Десятичный логарифм	Вычисляет десятичный логарифм вещественного числа.	стр. 2-240
EXP	Экспонента	Вычисляет значение экспоненциальной функции.	стр. 2-244
EXPT (**)	Возведение в степень	Возводит одно вещественное число в степень, равную другому вещественному числу.	стр. 2-247
Inc	Увеличение на единицу	Увеличивает целочисленное значение на единицу.	стр. 2-253
Dec	Уменьшение на единицу	Уменьшает целочисленное значение на единицу.	стр. 2-253

Команда	Имя	Функция	Стр.
Rand	Случайное число	Генерирует псевдослучайные числа.	стр. 2-255
AryAdd	Сложение массивов	Складывает соответствующие элементы двух массивов.	стр. 2-257
AryAddV	Прибавление значения к массиву	Прибавляет одно и то же значение к указанным элементам массива.	стр. 2-259
ArySub	Вычитание массивов	Вычитает соответствующие элементы двух массивов.	стр. 2-261
ArySubV	Вычитание значения из массива	Вычитает одно и то же значение из указанных элементов массива.	стр. 2-263
AryMean	Среднее значение массива	Вычисляет среднее значение элементов массива.	стр. 2-265
ArySD	Среднеквадратическое отклонение элементов массива	Вычисляет среднеквадратическое отклонение элементов массива.	стр. 2-267
ModReal	Вещественное деление по модулю	Вычисляет остаток от деления вещественных чисел.	стр. 2-269
Fraction	Дробная часть вещественного числа	Находит дробную часть вещественного числа.	стр. 2-272
CheckReal	Проверка вещественного числа	Проверяет, не является ли вещественное число бесконечностью или нечисловым значением.	стр. 2-274

Команды преобразования двоично-десятичных значений

Команда	Имя	Функция	Стр.
_BCD_TO_*	Группа преобразования BCD в целые числа без знака	Преобразует битовые строки в двоично-десятичном формате (BCD) в целые числа без знака.	стр. 2-278
_TO_BCD_*	Группа преобразования целых чисел без знака в BCD	Преобразует целые числа без знака в битовые строки в двоично-десятичном формате (BCD).	стр. 2-281
BCD_TO_**	Группа преобразования типа данных BCD в целые числа без знака	Преобразует битовые строки в двоично-десятичном формате (BCD) в целые числа без знака.	стр. 2-284
BCDsToBin	Преобразование BCD со знаком в целое число со знаком	Преобразует битовые строки в двоично-десятичном формате (BCD) со знаком в целые числа со знаком.	стр. 2-287
BinToBCDs_**	Группа преобразования целых чисел со знаком в BCD	Преобразует целые числа со знаком в битовые строки в двоично-десятичном формате (BCD) со знаком.	стр. 2-290
AryToBCD	Преобразование массива в BCD	Преобразует элементы массива, являющиеся целыми числами без знака, в битовые строки в двоично-десятичном формате (BCD).	стр. 2-293
AryToBin	Преобразование массива в беззнаковые целые	Преобразует элементы массива, являющиеся битовыми строками в двоично-десятичном формате (BCD), в целые числа без знака.	стр. 2-295

Команды преобразования типов данных

Команда	Имя	Функция	Стр.
_TO_* (Группа преобразования целых чисел в целые числа)	Группа преобразования целых чисел в целые числа	Преобразует целочисленные значения одного типа в целочисленные значения другого типа.	стр. 2-299
_TO_* (Группа преобразования целых чисел в битовые строки)	Группа преобразования целых чисел в битовые строки	Преобразует целочисленные значения в битовые строки.	стр. 2-302
_TO_* (Группа преобразования целых чисел в вещественные числа)	Группа преобразования целых чисел в вещественные числа	Преобразует целочисленные значения в вещественные значения.	стр. 2-305
_TO_* (Группа преобразования битовых строк в целые числа)	Группа преобразования битовых строк в целые числа	Преобразует битовые строки в целочисленные значения.	стр. 2-308
_TO_* (Группа преобразования битовых строк в битовые строки)	Группа преобразования битовых строк в битовые строки	Преобразует битовые строки одного типа в битовые строки другого типа.	стр. 2-311
_TO_* (Группа преобразования битовых строк в вещественные числа)	Группа преобразования битовых строк в вещественные числа	Преобразует битовые строки в вещественные значения.	стр. 2-313
_TO_* (Группа преобразования вещественных чисел в целые числа)	Группа преобразования вещественных чисел в целые числа	Преобразует вещественные значения в целочисленные значения.	стр. 2-315
_TO_* (Группа преобразования вещественных чисел в битовые строки)	Группа преобразования вещественных чисел в битовые строки	Преобразует вещественные значения в битовые строки.	стр. 2-318
_TO_* (Группа преобразования вещественных чисел в вещественные числа)	Группа преобразования вещественных чисел в вещественные числа	Преобразует вещественные числа одного типа в вещественные числа другого типа.	стр. 2-321
**_TO_STRING (Группа преобразования целых чисел в текстовые строки)	Группа преобразования целых чисел в текстовые строки	Преобразует целочисленные значения в текстовые строки.	стр. 2-323
**_TO_STRING (Группа преобразования битовых строк в текстовые строки)	Группа преобразования битовых строк в текстовые строки	Преобразует битовые строки в текстовые строки.	стр. 2-325
**_TO_STRING (Группа преобразования вещественных чисел в текстовые строки)	Группа преобразования вещественных чисел в текстовые строки	Преобразует вещественные значения в текстовые строки.	стр. 2-327
RealToFormatString	REAL в форматированную текстовую строку	Преобразует переменную типа REAL в текстовую строку с указанным форматом.	стр. 2-330

Команда	Имя	Функция	Стр.
LrealToFormatString	LREAL в форматированную текстовую строку	Преобразует переменную типа LREAL в текстовую строку с указанным форматом.	стр. 2-337
STRING_TO_** (Группа преобразования текстовых строк в целые числа)	Группа преобразования текстовых строк в целые числа	Преобразует текстовые строки в целочисленные значения.	стр. 2-344
STRING_TO_** (Группа преобразования текстовых строк в битовые строки)	Группа преобразования текстовых строк в битовые строки	Преобразует текстовые строки в битовые строки.	стр. 2-347
STRING_TO_** (Группа преобразования текстовых строк в вещественные числа)	Группа преобразования текстовых строк в вещественные числа	Преобразует текстовые строки в вещественные значения.	стр. 2-350
TO_** (Группа преобразования в целые числа)	Группа преобразования в целые числа	Преобразует целочисленные значения, битовые строки, вещественные значения и текстовые строки в целочисленные значения.	стр. 2-354
TO_** (Группа преобразования в битовые строки)	Группа преобразования в битовые строки	Преобразует целочисленные значения, битовые строки, вещественные значения и текстовые строки в битовые значения.	стр. 2-357
TO_** (Группа преобразования в вещественные числа)	Группа преобразования в вещественные числа	Преобразует целочисленные значения, битовые строки, вещественные значения и текстовые строки в вещественные значения.	стр. 2-359
EnumToNum	Перечисление в целое число	Преобразует значение перечислимого типа в значение типа DINT.	стр. 2-361
NumToEnum	Целое число в перечисление	Преобразует значение типа DINT в значение перечислимого типа.	стр. 2-363
TRUNC	Усечение	Усекает вещественное число до целого.	стр. 2-366
Round	Округление вещественного числа	Округляет вещественное число до ближайшего целого числа в большую или меньшую сторону в зависимости от значения разряда десятых.	стр. 2-366
RoundUp	Округление вещественного числа к большему	Округляет вещественное число до ближайшего целого числа в большую сторону.	стр. 2-366

Команды обработки битовых строк

Команда	Имя	Функция	Стр.
AND (&)	Логическое И	Выполняет побитовую операцию «логическое И» над несколькими переменными логического типа (BOOL) или битовыми строками.	стр. 2-370
OR	Логическое ИЛИ	Выполняет побитовую операцию «логическое ИЛИ» над несколькими переменными логического типа (BOOL) или битовыми строками.	стр. 2-370
XOR	Логическое исключающее ИЛИ	Выполняет побитовую операцию «логическое исключающее ИЛИ» над несколькими переменными логического типа (BOOL) или битовыми строками.	стр. 2-370
XORN	Логическое исключающее ИЛИ-НЕ	Выполняет побитовую операцию «логическое исключающее ИЛИ-НЕ» над несколькими переменными логического типа (BOOL) или битовыми строками.	стр. 2-373
NOT	Побитовая инверсия	Инвертирует каждый бит переменной логического типа (BOOL) или битовой строки.	стр. 2-375

Команда	Имя	Функция	Стр.
AryAnd	Логическое И массивов	Выполняет побитовую операцию «логическое И» над каждой парой элементов двух массивов, содержащих значения логического типа или битовые строки.	стр. 2-377
AryOr	Логическое ИЛИ массивов	Выполняет побитовую операцию «логическое ИЛИ» над каждой парой элементов двух массивов, содержащих значения логического типа или битовые строки.	стр. 2-377
AryXor	Логическое исключающее ИЛИ массивов	Выполняет побитовую операцию «логическое исключающее ИЛИ» над каждой парой элементов двух массивов, содержащих значения логического типа или битовые строки.	стр. 2-377
AryXorN	Логическое исключающее ИЛИ-НЕ массивов	Выполняет побитовую операцию «логическое исключающее ИЛИ-НЕ» над каждой парой элементов двух массивов, содержащих значения логического типа или битовые строки.	стр. 2-377

Команды выбора

Команда	Имя	Функция	Стр.
SEL	Двоичный выбор	Выбирает один из двух вариантов.	стр. 2-382
MUX	Мультиплексор	Производит выбор из нескольких (от двух до пяти) вариантов.	стр. 2-385
LIMIT	Ограничитель	Ограничивает значение входной переменной между указанными минимальным и максимальным значениями.	стр. 2-388
Band	Зона нечувствительности	Реализует зону нечувствительности.	стр. 2-390
Zone	Мертвая зона	Добавляет значение смещения к входному значению.	стр. 2-393
MAX	Максимум	Находит наибольшее из нескольких значений (от двух до пяти).	стр. 2-396
MIN	Минимум	Находит наименьшее из нескольких значений (от двух до пяти).	стр. 2-396
AryMax	Максимум массива	Находит элементы с наибольшим значением в одномерном массиве.	стр. 2-399
AryMin	Минимум массива	Находит элементы с наименьшим значением в одномерном массиве.	стр. 2-399
ArySearch	Поиск по массиву	Выполняет поиск указанного значения в одномерном массиве.	стр. 2-402

Команды перемещения данных

Команда	Имя	Функция	Стр.
MOVE	Перемещение	Перемещает значение константы или переменной в другую переменную.	стр. 2-406
MoveBit	Перемещение бита	Перемещает один бит в битовой строке.	стр. 2-409
MoveDigit	Перемещение разряда	Перемещает разряды (по 4 бита на разряд) в битовой строке.	стр. 2-411
TransBits	Перемещение битов	Перемещает один или несколько битов в битовой строке.	стр. 2-413
MemCopy	Копирование памяти	Перемещает один или несколько элементов массива. Источник перемещения и адресат перемещения должны иметь одинаковый тип данных.	стр. 2-416
SetBlock	Заполнение блока	Перемещает значение переменной или константы в один или несколько элементов массива.	стр. 2-418
Exchange	Обмен данными	Меняет местами значения двух переменных.	стр. 2-420

Команда	Имя	Функция	Стр.
AryExchange	Обмен данными массивов	Меняет местами элементы двух массивов.	стр. 2-422
AryMove	Перемещение массива	Перемещает один или несколько элементов массива. Тип данных исходного массива может отличаться от типа данных целевого массива.	стр. 2-424
Clear	Инициализация	Инициализирует переменную.	стр. 2-426
Copy**ToNum (Битовая строка в целое число со знаком)	Группа копирования комбинации битов (Битовая строка в целое число со знаком)	Копирует содержимое битовой строки непосредственно в целое число со знаком.	стр. 2-429
Copy**To*** (Битовая строка в вещественное число)	Группа копирования комбинации битов (Битовая строка в вещественное число)	Копирует содержимое битовой строки непосредственно в вещественное число.	стр. 2-431
CopyNumTo** (Целое число со знаком в битовую строку)	Группа копирования комбинации битов (Целое число со знаком в битовую строку)	Копирует содержимое целого числа со знаком непосредственно в битовую строку.	стр. 2-433
CopyNumTo** (Целое число со знаком в вещественное число)	Группа копирования комбинации битов (Целое число со знаком в вещественное число)	Копирует содержимое целого числа со знаком непосредственно в вещественное число.	стр. 2-435
Copy**To*** (Вещественное число в битовую строку)	Группа копирования комбинации битов (Вещественное число в битовую строку)	Копирует содержимое вещественного числа непосредственно в битовую строку.	стр. 2-437
Copy**ToNum (Вещественное число в целое число со знаком)	Группа копирования комбинации битов (Вещественное число в целое число со знаком)	Копирует содержимое вещественного числа непосредственно в целое число со знаком.	стр. 2-439

Команды сдвига

Команда	Имя	Функция	Стр.
AryShiftReg	Регистр сдвига	Сдвигает массив битовых строк на один бит влево и вставляет входное значение в самый младший значащий бит.	стр. 2-442
AryShiftRegLR	Реверсивный регистр сдвига	Сдвигает массив битовых строк на один бит влево или вправо и вставляет входное значение в самый младший или самый старший значащий бит.	стр. 2-445
ArySHL	Сдвиг массива влево на N элементов	Сдвигает массив влево на один или несколько элементов (в сторону элементов с большими индексами).	стр. 2-448
ArySHR	Сдвиг массива вправо на N элементов	Сдвигает массив вправо на один или несколько элементов (в сторону элементов с меньшими индексами).	стр. 2-448
SHL	Сдвиг влево на N битов	Сдвигает битовую строку влево на один или несколько битов (в сторону старших битов).	стр. 2-451
SHR	Сдвиг вправо на N битов	Сдвигает битовую строку вправо на один или несколько битов (в сторону младших битов).	стр. 2-451

Команда	Имя	Функция	Стр.
NSHLC	Сдвиг на N битов влево с переносом	Сдвигает массив битовых строк на один или несколько битов влево (в сторону элементов с большими индексами) с использованием флага переноса (CY).	стр. 2-454
NSHRC	Сдвиг на N битов вправо с переносом	Сдвигает массив битовых строк на один или несколько битов вправо (в сторону элементов с меньшими индексами) с использованием флага переноса (CY).	стр. 2-454
ROL	Циклический сдвиг на N битов влево	Циклически сдвигает битовую строку влево на один или несколько битов (в сторону старших битов).	стр. 2-457
ROR	Циклический сдвиг на N битов вправо	Циклически сдвигает битовую строку вправо на один или несколько битов (в сторону младших битов).	стр. 2-457

Команды преобразования

Команда	Имя	Функция	Стр.
Swap	Перестановка байтов	Меняет местами старший и младший байты 16-битного значения.	стр. 2-463
Neg	Инверсия знака	Меняет знак числа на противоположный.	стр. 2-465
Decoder	Битовый дешифратор	Устанавливает указанный бит в состояние ИСТИНА, а остальные биты сбрасывает в состояние ЛОЖЬ в массиве элементов, состоящем максимум из 256 битов.	стр. 2-467
Encoder	Битовый шифратор	Определяет позицию самого старшего бита в состоянии ИСТИНА в элементах массива, содержащего максимум 256 битов.	стр. 2-470
BitCnt	Подсчет битов	Подсчитывает количество битов в состоянии ИСТИНА в битовой строке.	стр. 2-472
ColmToLine_**	Группа преобразования столбца в строку	Извлекает значения битов в указанной позиции каждого элемента массива и выводит их в виде битовой строки.	стр. 2-474
LineToColm	Преобразование строки в столбец	Извлекает биты из битовой строки и выводит каждый из них в соответствующий бит в указанной позиции каждого элемента массива.	стр. 2-476
Gray	Преобразование кода Грея	Преобразует код Грея в значение угла.	стр. 2-479
UTF8ToSJIS	Преобразование кодировки UTF-8 в SJIS	Преобразует текстовую строку в кодировке UTF-8 в массив байтов (BYTE) с кодами символов в кодировке SJIS.	стр. 2-484
SJISToUTF8	Преобразование кодировки SJIS в UTF-8	Преобразует массив байтов, содержащих коды символов в кодировке SJIS, в текстовую строку в кодировке UTF-8.	стр. 2-486
PWLApprox	Аппроксимация ломаной линии с проверкой данных ломаной линии	Аппроксимирует целочисленные или вещественные координаты ломаной линии и проверяет достоверность данных ломаной линии.	стр. 2-488
PWLApproxNoLineChk	Аппроксимация ломаной линии без проверки данных ломаной линии	Аппроксимирует целочисленные или вещественные координаты ломаной линии без проверки достоверности данных ломаной линии.	стр. 2-488
PWLLineChk	Проверка данных ломаной линии	Проверяет, упорядочены ли данные ломаной линии, используемые для команды PWLApproxNoLineCheck, в порядке возрастания значений координаты X.	стр. 2-494
MovingAverage	Скользящее среднее	Вычисляет скользящее среднее значение.	стр. 2-497
DispartReal	Разделение на мантиссу и показатель степени	Разделяет вещественное число на мантиссу со знаком и показатель степени.	стр. 2-504

Команда	Имя	Функция	Стр.
UniteReal	Формирование вещественного числа из мантиссы и показателя степени	Объединяет мантиссу со знаком и показатель степени и формирует из них вещественное число.	стр. 2-507
NumToDecString	Преобразование в текстовую строку фиксированной длины в десятичном виде	Преобразует целое число в текстовую строку фиксированной длины, содержащую число в десятичном виде.	стр. 2-509
NumToHexString	Преобразование в текстовую строку фиксированной длины в шестнадцатеричном виде	Преобразует целое число в текстовую строку фиксированной длины, содержащую число в шестнадцатеричном виде.	стр. 2-509
HexStringToNum_**	Группа преобразования текстовой строки с шестнадцатеричными цифрами в число	Преобразует текстовую строку, содержащую шестнадцатеричные цифры, в эквивалентное целое число.	стр. 2-513
FixNumToString	Преобразование числа с фиксированной запятой в текстовую строку	Преобразует число с фиксированной запятой со знаком в текстовую строку, содержащую представление этого числа в десятичном виде.	стр. 2-515
StringToFixNum	Преобразование текстовой строки в число с фиксированной запятой	Преобразует текстовую строку с десятичным числом в число с фиксированной запятой со знаком.	стр. 2-517
DtToString	Преобразование даты и времени в текстовую строку	Преобразует дату и время в текстовую строку.	стр. 2-520
DateToString	Преобразование даты в текстовую строку	Преобразует дату в текстовую строку.	стр. 2-522
TodToString	Преобразование времени суток в текстовую строку	Преобразует время суток в текстовую строку.	стр. 2-524
GrayToBin_**	Группа преобразования кода Грея в двоичный код	Преобразует код Грея в битовую строку.	стр. 2-526
BinToGray_**	Преобразование двоичного кода в код Грея	Преобразует битовую строку в код Грея.	стр. 2-526
StringToAry	Преобразование текстовой строки в массив	Преобразует текстовую строку в массив байтов (BYTE).	стр. 2-529
AryToString	Преобразование массива в текстовую строку	Преобразует массив байтов (BYTE) в текстовую строку.	стр. 2-531
DispartDigit	Разделение на 4-битные блоки	Разделяет битовую строку на 4-битные блоки.	стр. 2-533
UniteDigit_**	Группа объединения 4-битных блоков	Объединяет 4-битные блоки данных в битовую строку.	стр. 2-535
Dispart8Bit	Разделение данных на байты	Разделяет битовую строку на отдельные байты.	стр. 2-537
Unite8Bit_**	Группа объединения байтов данных	Объединяет байты данных в битовую строку.	стр. 2-539

Команда	Имя	Функция	Стр.
ToAryByte	Преобразование в байтовый массив	Разделяет переменную на байты и сохраняет их в массив типа BYTE.	стр. 2-541
AryByteTo	Преобразование из байтового массива	Объединяет элементы массива типа BYTE и сохраняет результат в переменную.	стр. 2-547
SizeOfAry	Получить количество элементов массива	Позволяет узнать количество элементов в массиве.	стр. 2-554
PackWord	Объединение двух байтов	Объединяет два однобайтных значения в двухбайтное значение.	стр. 2-556
PackDword	Объединение четырех байтов	Объединяет четыре однобайтных значения в четырехбайтное значение.	стр. 2-558
LOWER_BOUND	Получить минимальное значение индекса массива	Получает минимальное значение индекса для указанного измерения массива.	стр. 2-560
UPPER_BOUND	Получить максимальное значение индекса массива	Получает максимальное значение индекса для указанного измерения массива.	стр. 2-560

Команды для работы со стеком и таблицами

Команда	Имя	Функция	Стр.
StackPush	Ввод в стек	Сохраняет значение на вершину стека.	стр. 2-566
StackFIFO	Первым вошел — первым вышел	Удаляет самое нижнее значение стека.	стр. 2-575
StackLIFO	Последним вошел — первым вышел	Удаляет самое верхнее значение стека.	стр. 2-575
StackIns	Вставка в стек	Вставляет значение в указанную позицию стека.	стр. 2-579
StackDel	Удаление из стека	Удаляет значение из указанной позиции в стеке.	стр. 2-582
RecSearch	Поиск записи	Производит поиск элементов, содержащих заданное значение, среди элементов массива структур, используя указанный метод поиска.	стр. 2-584
RecRangeSearch	Поиск диапазона записей	Производит поиск в массиве структур, используя указанный метод поиска. Ищутся элементы, содержащие значение, которое находится в заданном диапазоне.	стр. 2-589
RecSort	Сортировка записей	Сортирует элементы массива структур.	стр. 2-594
RecNum	Получить количество записей	Определяет количество записей в массиве структур, предшествующих искомым («конечным») данным.	стр. 2-601
RecMax	Поиск записи с максимальным значением	Ищет в массиве структур максимальное значение указанного члена структуры.	стр. 2-604
RecMin	Поиск записи с минимальным значением	Ищет в массиве структур минимальное значение указанного члена структуры.	стр. 2-604

Команды вычисления контрольной суммы

Команда	Имя	Функция	Стр.
StringSum	Расчет контрольной суммы	Вычисляет контрольную сумму для текстовой строки.	стр. 2-610
StringLRC	Расчет LRC текстовой строки	Вычисляет значение LRC (поперечный контроль четности).	стр. 2-612
StringCRCCCITT	Расчет CRC-CCITT текстовой строки	Вычисляет значение CRC-CCITT, используя метод XMODEM.	стр. 2-614

Команда	Имя	Функция	Стр.
StringCRC16	Расчет CRC-16 текстовой строки	Вычисляет значение CRC-16, используя метод MODBUS.	стр. 2-616
AryLRC_**	Группа расчета LRC массива	Вычисляет значение LRC массива.	стр. 2-618
AryCRCCCITT	Расчет CRC-CCITT массива	Вычисляет значение CRC-CCITT, используя метод XMODEM.	стр. 2-620
AryCRC16	Расчет CRC-16 массива	Вычисляет значение CRC-16, используя метод MODBUS.	стр. 2-622

Команды для обработки текстовых строк

Команда	Имя	Функция	Стр.
CONCAT	Конкатенация строк	Объединяет от двух до пяти текстовых строк.	стр. 2-626
LEFT	Получить строку слева	Извлекает подстроку с указанным количеством символов, расположенную в начале (слева) текстовой строки.	стр. 2-628
RIGHT	Получить строку справа	Извлекает подстроку с указанным количеством символов, расположенную в конце (справа) текстовой строки.	стр. 2-628
MID	Получить строку в указанной позиции	Извлекает подстроку с указанным количеством символов, расположенную в указанной позиции текстовой строки.	стр. 2-631
FIND	Поиск строки	Определяет позицию указанной подстроки в текстовой строке.	стр. 2-633
LEN	Длина строки	Определяет количество символов в текстовой строке.	стр. 2-635
REPLACE	Замена строки	Заменяет часть текстовой строки другой текстовой строкой.	стр. 2-637
DELETE	Удаление строки	Удаляет всю текстовую строку или ее часть.	стр. 2-639
INSERT	Вставка строки	Вставляет текстовую строку в другую текстовую строку.	стр. 2-641
GetByteLen	Получить длину в байтах	Подсчитывает количество байтов в текстовой строке.	стр. 2-643
ClearString	Очистка строки	Очищает текстовую строку.	стр. 2-645
ToUCase	Перевод в верхний регистр	Переводит все однобайтовые буквы в текстовой строке в верхний регистр.	стр. 2-647
ToLCase	Перевод в нижний регистр	Переводит все однобайтовые буквы в текстовой строке в нижний регистр.	стр. 2-647
TrimL	Обрезка строки слева	Удаляет пробел в начале текстовой строки.	стр. 2-649
TrimR	Обрезка строки справа	Удаляет пробел в конце текстовой строки.	стр. 2-649
AddDelimiter	Поместить в текстовую строку с разделителями	Преобразует значения всех членов в структуре в текстовую строку с разделителями.	стр. 2-651
SubDelimiter	Получить текстовые строки без разделителей	Считывает разделенные части текстовой строки и сохраняет их как значения членов структуры.	стр. 2-664

Команды для обработки времени и времени суток

Команда	Имя	Функция	Стр.
ADD_TIME	Сложение времени	Складывает два значения времени.	стр. 2-679
ADD_TOD_TIME	Сложение времени с временем суток	Складывает значение времени со значением времени суток.	стр. 2-681
ADD_DT_TIME	Сложение времени с датой и временем	Складывает значение времени со значением даты и времени.	стр. 2-683

Команда	Имя	Функция	Стр.
SUB_TIME	Вычитание времени	Вычитает значение времени из другого значения времени.	стр. 2-685
SUB_TOD_TIME	Вычитание времени из времени суток	Вычитает значение времени из значения времени суток.	стр. 2-687
SUB_TOD_TOD	Вычитание времени суток	Вычитает значение времени суток из другого значения времени суток.	стр. 2-689
SUB_DATE_DATE	Вычитание даты	Вычитает значение даты из другого значения даты.	стр. 2-691
SUB_DT_DT	Вычитание даты и времени	Вычитает значение даты и времени из другого значения даты и времени.	стр. 2-693
SUB_DT_TIME	Вычитание времени из даты и времени	Вычитает значение времени из значения даты и времени.	стр. 2-695
MULTIME	Умножение времени	Умножает значение времени на заданное число.	стр. 2-697
DIVTIME	Деление времени	Делит значение времени на заданное число.	стр. 2-699
CONCAT_DATE_TOD	Объединение даты и времени суток	Объединяет значение даты со значением времени суток.	стр. 2-701
DT_TO_TOD	Извлечение времени суток из даты и времени	Извлекает значение времени суток из значения даты и времени.	стр. 2-703
DT_TO_DATE	Извлечение даты из даты и времени	Извлекает значение даты из значения даты и времени.	стр. 2-705
SetTime	Установка времени	Устанавливает системное время.	стр. 2-707
GetTime	Получить время суток	Считывает текущее время.	стр. 2-709
DtToSec	Преобразование даты и времени в секунды	Преобразует дату и время в количество секунд, начиная с 00:00:00, 1 января 1970 г.	стр. 2-711
DateToSec	Преобразование даты в секунды	Преобразует дату в количество секунд, начиная с 00:00:00, 1 января 1970 г.	стр. 2-713
TodToSec	Преобразование времени суток в секунды	Преобразует время суток в количество секунд, начиная с 00:00:00.	стр. 2-715
SecToDt	Преобразование секунд в дату и время	Преобразует количество секунд, начиная с 00:00:00, 1 января 1970 г., в значение даты и времени.	стр. 2-717
SecToDate	Преобразование секунд в дату	Преобразует количество секунд, начиная с 00:00:00, 1 января 1970 г., в значение даты.	стр. 2-719
SecToTod	Преобразование секунд в время суток	Преобразует количество секунд, начиная с 00:00:00, в значение времени суток.	стр. 2-721
TimeToNanoSec	Преобразование времени в наносекунды	Преобразует значение времени в наносекунды.	стр. 2-723
TimeToSec	Преобразование времени в секунды	Преобразует значение времени в секунды.	стр. 2-725
NanoSecToTime	Преобразование наносекунд во время	Преобразует количество наносекунд в значение времени.	стр. 2-727
SecToTime	Преобразование секунд во время	Преобразует количество секунд в значение времени.	стр. 2-729
ChkLeapYear	Проверка на високосный год	Проверяет, является ли указанный год високосным.	стр. 2-731

Команда	Имя	Функция	Стр.
GetDaysOfMonth	Получить количество дней в месяце	Позволяет узнать количество дней в указанном месяце.	стр. 2-733
DaysToMonth	Преобразование дней в месяц	Вычисляет месяц на основании количества дней, начиная с 1 января.	стр. 2-736
GetDayOfWeek	Получить день недели	Позволяет определить день недели для указанной даты (года, месяца и дня).	стр. 2-738
GetWeekOfYear	Получить номер недели	Позволяет определить номер недели для указанной даты (года, месяца и дня).	стр. 2-740
DtToDateStruct	Разбивка даты и времени	Преобразует дату и время в год, месяц, день, час, минуты, секунды и наносекунды.	стр. 2-742
DateStructToDt	Объединение времени	Объединяет значения года, месяца, дня, часа, минут, секунд и наносекунд в значение даты и времени.	стр. 2-745
TruncTime	Усечение времени	Усекает переменную типа TIME до заданной единицы измерения времени.	стр. 2-748
TruncDt	Усечение даты и времени	Усекает переменную типа DT до заданной единицы измерения времени.	стр. 2-753
TruncTod	Усечение времени суток	Усекает переменную типа TOD до заданной единицы измерения времени.	стр. 2-757

Команды для аналогового регулирования

Команда	Имя	Функция	Стр.
PIDAT	ПИД-регулятор с автонастройкой	ПИД-регулятор с автонастройкой (2-ПИД-регулятор с фильтром уставки).	стр. 2-762
PIDAT_HeatCool	ПИД-регулятор нагрева/охлаждения с автонастройкой	Реализует ПИД-регулятор нагрева/охлаждения с автонастройкой (2-ПИД-регулятор с фильтром уставки).	стр. 2-796
TimeProportionalOut	Выход широтно-импульсного регулирования	Преобразует управляющее воздействие в выходной сигнал для широтно-импульсного регулирования (ШИР).	стр. 2-839
LimitAlarm_**	Группа сигнализации аварии выхода за верхний/нижний предел	Выдает сигнал аварии, если входное значение меньше установленного нижнего предельного значения или больше установленного верхнего предельного значения.	стр. 2-860
LimitAlarmDv_**	Группа сигнализации аварии верхнего/нижнего отклонения	Выдает сигнал аварии, если отклонение входного значения от опорного значения превышает заданное значение нижнего отклонения или заданное значение верхнего отклонения.	стр. 2-865
LimitAlarmDvStbySeq_**	Группа сигнализации аварии верхнего/нижнего отклонения с контролем последовательности	Выдает сигналы аварии по нижнему и верхнему отклонению с контролем последовательности событий.	стр. 2-871
ScaleTrans	Изменение масштаба	Преобразует входное значение в выходное значение пропорционально соотношению входного и выходного диапазонов.	стр. 2-891
AC_StepProgram	Поэтапная программа	Производит расчет текущего заданного значения и прогнозируемого заданного значения в каждом периоде выполнения задачи в соответствии с заданной программной последовательностью.	стр. 2-894

Команды для управления системой

Команда	Имя	Функция	Стр.
TraceSamp	Отбор данных для протокола данных	Выполняет отбор данных для протокола данных.	стр. 2-925
TraceTrig	Запуск протокола данных	Активирует событие запуска для протоколирования данных.	стр. 2-929
GetTraceStatus	Чтение состояния протокола данных	Считывает состояние выполнения протокола данных.	стр. 2-932
SetAlarm	Создание ошибки пользователя	Создает ошибку пользователя.	стр. 2-936
ResetAlarm	Сброс ошибки пользователя	Сбрасывает ошибку пользователя.	стр. 2-941
GetAlarm	Получить состояние ошибки пользователя	Возвращает наивысший уровень события (среди уровней ошибки пользователя 1...8) и код события наивысшего уровня среди текущих ошибок пользователя.	стр. 2-943
ResetPLCError	Сброс ошибки контроллера в PLC	Сбрасывает ошибки в функциональном модуле «PLC».	стр. 2-945
GetPLCError	Получить состояние ошибки контроллера в PLC	Определяет наиболее высокий уровень события (частично критическая ошибка или некритическая ошибка) и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «PLC».	стр. 2-949
ResetCJBError	Сброс ошибки контроллера в шине CJ	Сбрасывает ошибки контроллера в шине ввода-вывода.	стр. 2-951
GetCJBError	Получить состояние ошибки шины ввода-вывода	Определяет наиболее высокий уровень события и код события наиболее высокого уровня среди текущих ошибок контроллера в шине ввода-вывода модуля ЦПУ серии NJ.	стр. 2-953
GetEIPError	Получить состояние ошибки в EtherNet/IP	Определяет наиболее высокий уровень события (частично критическая ошибка или некритическая ошибка) и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «EtherNet/IP».	стр. 2-955
ResetMCError	Сброс ошибки в Motion Control	Сбрасывает ошибки контроллера в функциональном модуле «Motion Control».	стр. 2-957
GetMCError	Получить состояние ошибки в Motion Control	Определяет наиболее высокий уровень события (частично критическая ошибка или некритическая ошибка) и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «Motion Control».	стр. 2-963
ResetECError	Сброс ошибки в EtherCAT	Сбрасывает ошибки контроллера в функциональном модуле «EtherCAT Master».	стр. 2-966
GetECError	Получить состояние ошибки в EtherCAT	Обнаруживает ошибки в функциональном модуле «EtherCAT Master».	стр. 2-968
ResetNXBError	Сброс ошибки в NX Bus	Сбрасывает ошибки контроллера в функциональном модуле «NX Bus».	стр. 2-971
GetNXBError	Получить состояние ошибки в NX Bus	Определяет наиболее высокий уровень события среди текущих ошибок контроллера в функциональном модуле «NX Bus» модуля ЦПУ серии NX.	стр. 2-973
GetNXUnitError	Получить состояние ошибки в модуле NX	Определяет наиболее высокий уровень события и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «NX Bus» модуля ЦПУ серии NX или модулей NX.	стр. 2-975

Команда	Имя	Функция	Стр.
SetInfo	Создание информации пользователя	Создает информацию пользователя.	стр. 2-983
ResetUnit	Перезапустить модуль	Перезапускает модуль шины ЦПУ или специальный модуль ввода-вывода.	стр. 2-985
GetNTPStatus	Чтение состояния NTP	Производит чтение состояния NTP.	стр. 2-990
RestartNXUnit	Перезапуск модулей NX	Перезапускает интерфейсный модуль EtherCAT или модули NX.	стр. 2-992
NX_ChangeWriteMode	Перевод модуля NX в режим записи	Переводит интерфейсный модуль EtherCAT или модуль NX в режим, в котором разрешена запись данных.	стр. 2-998
NX_SaveParam	Сохранение параметров модуля NX	Сохраняет данные, которые были записаны в интерфейсный модуль EtherCAT или модуль NX.	стр. 2-1004
PLC_ReadTotalPowerOnTime	Чтение общего времени работы ПЛК	Считывает общее время работы при включенном питании из указанного модуля ЦПУ.	стр. 2-1010
NX_ReadTotalPowerOnTime	Чтение общего времени работы модуля NX	Считывает общее время работы при включенном питании из интерфейсного модуля или модуля NX.	стр. 2-1013

Команды для управления программами

Команда	Имя	Функция	Стр.
PrgStart	Активация программы	Разрешает выполнение указанной программы.	стр. 2-1022
PrgStop	Деактивация программы	Запрещает выполнение указанной программы.	стр. 2-1032
PrgStatus	Чтение статуса программы	Производит чтение статуса указанной программы.	стр. 2-1052

Команды для обмена данными по интерфейсу EtherCAT

Команда	Имя	Функция	Стр.
EC_CoESDOWrite	Запись в SDO CoE в EtherCAT	Производит запись значения в объект CoE указанного ведомого устройства в сети EtherCAT.	стр. 2-1060
EC_CoESDORead	Чтение из SDO CoE в EtherCAT	Производит чтение значения из объекта CoE указанного ведомого устройства в сети EtherCAT.	стр. 2-1064
EC_StartMon	Запуск мониторинга пакетов EtherCAT	Запускает мониторинг пакетов для интерфейса связи EtherCAT.	стр. 2-1070
EC_StopMon	Остановка мониторинга пакетов EtherCAT	Останавливает выполнение мониторинга пакетов для интерфейса связи EtherCAT.	стр. 2-1076
EC_SaveMon	Сохранение пакетов EtherCAT	Сохраняет данные коммуникационных пакетов EtherCAT во внутренний файл в оперативной памяти модуля ЦПУ.	стр. 2-1078
EC_CopyMon	Передача пакетов EtherCAT	Передает данные пакетов во внутреннем файле в оперативной памяти модуля ЦПУ на карту памяти SD.	стр. 2-1080
EC_DisconnectSlave	Отсоединение ведомого устройства EtherCAT	Отсоединяет указанное ведомое устройство от сети EtherCAT.	стр. 2-1082
EC_ConnectSlave	Подсоединение ведомого устройства EtherCAT	Подсоединяет указанное ведомое устройство к сети EtherCAT.	стр. 2-1091

Команда	Имя	Функция	Стр.
EC_ChangeEnableSetting	Активация/деактивация ведомого устройства EtherCAT	Активирует или деактивирует ведомое устройство EtherCAT.	стр. 2-1093
NX_WriteObj	Запись в объект модуля NX	Производит запись данных в объект NX в интерфейсном модуле EtherCAT или модуле NX.	стр. 2-1114
NX_ReadObj	Чтение объекта модуля NX	Производит чтение данных из объекта NX в интерфейсном модуле EtherCAT или модуле NX.	стр. 2-1131

Команды для обмена данными по интерфейсу IO-Link

Команда	Имя	Функция	Стр.
IOL_ReadObj	Чтение из объекта устройства IO-Link	Производит чтение данных из объектов устройств IO-Link.	стр. 2-1142
IOL_WriteObj	Запись в объект устройства IO-Link	Производит запись данных в объекты устройств IO-Link.	стр. 2-1151

Команды для обмена данными по интерфейсу EtherNet/IP

Команда	Имя	Функция	Стр.
CIPOpen	Открытие CIP-соединения класса 3 (Large_Forward_Open)	Открывает соединение класса 3 (Large_Forward_Open) по протоколу CIP с указанным удаленным узлом. Длина данных устанавливается равной 1994 байт.	стр. 2-1163
CIPOpenWithDataSize	Открытие CIP-соединения класса 3 с указанным объемом данных	Открывает соединение класса 3 по протоколу CIP с указанным удаленным узлом, которое позволяет передавать и принимать явные сообщения класса 3 с указанной длиной данных или меньшей длиной.	стр. 2-1174
CIPRead	Чтение переменной посредством явного сообщения класса 3	Использует явное сообщение класса 3 для чтения значения переменной в другом контроллере в сети CIP.	стр. 2-1178
CIPWrite	Запись переменной посредством явного сообщения класса 3	Использует явное сообщение класса 3 для записи значения в переменную в другом контроллере в сети CIP.	стр. 2-1184
CIPSend	Передача явного сообщения класса 3	Передает сообщение класса 3 по протоколу CIP указанному устройству в сети CIP.	стр. 2-1191
CIPClose	Закрытие CIP-соединения класса 3	Закрывает соединение класса 3 по протоколу CIP для указанного дескриптора.	стр. 2-1197
CIPUCMMRead	Чтение переменной посредством явного сообщения UCMM	Использует явное сообщение UCMM для чтения значения переменной в другом контроллере в указанной сети CIP.	стр. 2-1200
CIPUCMMWrite	Запись переменной посредством явного сообщения UCMM	Использует явное сообщение UCMM для записи значения в переменную в другом контроллере в сети CIP.	стр. 2-1206
CIPUCMMSend	Передача явного сообщения UCMM	Передает сообщение UCMM по протоколу CIP указанному устройству в сети CIP.	стр. 2-1214
SktUDPCreate	Создание сокета UDP	Создает запрос на открытие сокета UDP для открытия порта UDP на встроенном порте EtherNet/IP.	стр. 2-1227

Команда	Имя	Функция	Стр.
SktUDPRcv	Прием через сокет UDP	Считывает данные из буфера приема для сокета UDP на встроенном порте EtherNet/IP.	стр. 2-1236
SktUDPSend	Передача через сокет UDP	Передает данные с порта UDP на встроенном порте EtherNet/IP.	стр. 2-1240
SktTCPAccept	Принятие сокета TCP	Запрашивает принятие сокета TCP для встроенного порта EtherNet/IP.	стр. 2-1243
SktTCPConnect	Соединение с сокетом TCP	Устанавливает соединение с удаленным портом TCP через встроенный порт EtherNet/IP.	стр. 2-1246
SktTCPRcv	Прием через сокет TCP	Считывает данные из буфера приема для сокета TCP на встроенном порте EtherNet/IP.	стр. 2-1255
SktTCPSend	Передача через сокет TCP	Передает данные с порта TCP на встроенном порте EtherNet/IP.	стр. 2-1259
SktGetTCPStatus	Чтение состояния сокета TCP	Производит чтение состояния сокета TCP.	стр. 2-1262
SktClose	Закрытие сокета TCP/UDP	Закрывает указанный сокет TCP или UDP для встроенного порта EtherNet/IP.	стр. 2-1265
SktClearBuf	Очистка буфера приема сокета TCP/UDP	Очищает буфер приема для указанного сокета TCP или UDP для встроенного порта EtherNet/IP.	стр. 2-1268
SktSetOption	Настройка дополнительного параметра сокета TCP	Позволяет настроить дополнительный параметр для указанного сокета TCP для встроенного порта EtherNet/IP.	стр. 2-1271
ModbusTCPCmd	Передача команды общего назначения по протоколу Modbus TCP	Передает команды общего назначения с использованием протокола Modbus-TCP.	стр. 2-1277
ModbusTCPRead	Передача команды чтения по протоколу Modbus TCP	Производит чтение данных, которые запрашиваются путем передачи команд чтения с использованием протокола Modbus-TCP.	стр. 2-1286
ModbusTCPWrite	Передача команды записи по протоколу Modbus TCP	Передает команды записи с использованием протокола Modbus-TCP.	стр. 2-1295
ChangeIPAdr	Изменение IP-адреса	Изменяет IP-адрес встроенного порта EtherNet/IP модуля ЦПУ или IP-адрес модуля интерфейса EtherNet/IP.	стр. 2-1303
ChangeFTPAccount	Изменение учетной записи FTP	Изменяет имя и пароль для входа на сервер FTP для встроенного порта EtherNet/IP модуля ЦПУ или для модуля интерфейса EtherNet/IP.	стр. 2-1312
ChangeNTPServerAdr	Изменение адреса сервера NTP	Изменяет адрес сервера NTP встроенного порта EtherNet/IP модуля ЦПУ или адрес сервера NTP модуля интерфейса EtherNet/IP.	стр. 2-1316
FTPGetFileList	Получение списка файлов на сервере FTP	Позволяет получить список файлов, находящихся на сервере FTP.	стр. 2-1321
FTPGetFile	Получить файл с сервера FTP	Загружает файл с сервера FTP.	стр. 2-1337
FTPPutFile	Поместить файл на сервер FTP	Загружает файл на сервер FTP.	стр. 2-1347
FTPRemoveFile	Удаление файла с сервера FTP	Удаляет файл с сервера FTP.	стр. 2-1358
FTPRemoveDir	Удаление каталога с сервера FTP	Удаляет каталог с сервера FTP.	стр. 2-1369

Команды для обмена данными по последовательному интерфейсу

Команда	Имя	Функция	Стр.
ExecPMCR	Макрос протокола	Запрашивает выполнение коммуникационной последовательности (макроса протокола), зарегистрированной в модуле последовательного интерфейса.	стр. 2-1376
SerialSend	Передача через последовательный порт SCU	Выполняет передачу данных в беспротокольном режиме через последовательный порт модуля последовательного интерфейса.	стр. 2-1391
SerialRcv	Прием через последовательный порт SCU	Выполняет прием данных в беспротокольном режиме через последовательный порт модуля последовательного интерфейса. Очищает буфер приема после чтения данных.	стр. 2-1403
SerialRcvNoClear	Прием через последовательный порт SCU без очистки буфера приема	Выполняет прием данных в беспротокольном режиме через последовательный порт модуля последовательного интерфейса. Не очищает буфер приема после чтения данных.	стр. 2-1403
SendCmd	Передача команды	Служит для передачи команды модулю последовательного интерфейса с использованием шлюза последовательного интерфейса. Также позволяет передать в явном виде команду модулю сети DeviceNet или модулю ведущего устройства сети CompoNet.	стр. 2-1419
NX_SerialSend	Передача данных в беспротокольном режиме	Выполняет передачу данных в беспротокольном режиме через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы.	стр. 2-1432
NX_SerialRcv	Прием данных в беспротокольном режиме	Выполняет чтение данных в беспротокольном режиме через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы.	стр. 2-1445
NX_ModbusRtuCmd	Передача команды общего назначения по протоколу Modbus RTU	Передает команды общего назначения через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы ведомым устройствам Modbus-RTU с использованием протокола Modbus-RTU.	стр. 2-1460
NX_ModbusRtuRead	Передача команды чтения по протоколу Modbus RTU	Передает команды чтения через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы ведомым устройствам Modbus-RTU с использованием протокола Modbus-RTU.	стр. 2-1472
NX_ModbusRtuWrite	Передача команды записи по протоколу Modbus RTU	Передает команды записи через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы ведомым устройствам Modbus-RTU с использованием протокола Modbus-RTU.	стр. 2-1485
NX_SerialSigCtl	Включение/выключение сигнала управления последовательного порта	Включает или выключает сигнал ER или RS последовательного порта интерфейсного модуля серии NX или последовательного порта дополнительной платы.	стр. 2-1498
NX_SerialSigRead	Чтение сигнала управления последовательного порта	Считывает состояние сигнала CS или DR последовательного порта дополнительной платы.	стр. 2-1506
NX_SerialStatusRead	Чтение состояния последовательного порта	Считывает состояние последовательного порта дополнительной платы.	стр. 2-1511
NX_SerialBufClear	Очистка буфера	Очищает буфер передачи или приема.	стр. 2-1516

Команда	Имя	Функция	Стр.
NX_SerialStartMon	Запуск мониторинга линии последовательного интерфейса	Запускает мониторинг линии последовательного интерфейса интерфейсного модуля серии NX.	стр. 2-1527
NX_SerialStopMon	Остановка мониторинга линии последовательного интерфейса	Останавливает мониторинг линии последовательного интерфейса интерфейсного модуля серии NX.	стр. 2-1532

Команды для работы с картой памяти SD

Команда	Имя	Функция	Стр.
FileWriteVar	Запись переменной в файл	Записывает значение переменной в указанный файл на карте памяти SD. Значение записывается в двоичном формате.	стр. 2-1538
FileReadVar	Чтение переменной из файла	Считывает содержимое указанного файла на карте памяти SD как двоичные данные и записывает его в переменную.	стр. 2-1544
FileOpen	Открытие файла	Открывает указанный файл на карте памяти SD.	стр. 2-1550
FileClose	Закрытие файла	Закрывает указанный файл на карте памяти SD.	стр. 2-1554
FileSeek	Поиск в файле	Устанавливает указатель позиции на требуемую позицию в указанном файле на карте памяти SD.	стр. 2-1557
FileRead	Чтение из файла	Считывает данные из указанного файла на карте памяти SD.	стр. 2-1560
FileWrite	Запись в файл	Записывает данные в указанный файл на карте памяти SD.	стр. 2-1569
FileGets	Получить текстовую строку	Считывает текстовую строку из одной строки указанного файла на карте памяти SD.	стр. 2-1577
FilePuts	Поместить текстовую строку	Записывает текстовую строку в указанный файл на карте памяти SD.	стр. 2-1585
FileCopy	Копирование файла	Создает копию указанного файла на карте памяти SD.	стр. 2-1594
FileRemove	Удаление файла	Удаляет указанный файл с карты памяти SD.	стр. 2-1602
FileRename	Изменение имени файла	Изменяет имя указанного файла или каталога на карте памяти SD.	стр. 2-1607
DirCreate	Создание каталога	Создает каталог с указанным именем на карте памяти SD.	стр. 2-1613
DirRemove	Удаление каталога	Удаляет указанный каталог с карты памяти SD.	стр. 2-1616
BackupToMemoryCard	Резервное копирование на карту памяти SD	Создает резервную копию данных на карте памяти SD.	стр. 2-1620

Команды с использованием меток времени

Команда	Имя	Функция	Стр.
NX_DOutTimeStamp	Запись в дискретный выход с указанием метки времени	Записывает значение в выходной бит модуля дискретных выходов, поддерживающего обновление с использованием меток времени.	стр. 2-1636
NX_AryDOutTimeStamp	Запись в дискретный выход из массива с указанием метки времени	Выдает последовательность импульсов с выхода модуля дискретных выходов, поддерживающего обновление с указанием метки времени.	стр. 2-1643

Прочие команды

Команда	Имя	Функция	Стр.
ReadNbit_**	Группа чтения N битов	Считывает состояния ноля или большего числа битов в битовой строке.	стр. 2-1654
WriteNbit_**	Группа записи N битов	Производят запись в ноль или большее число битов в битовой строке.	стр. 2-1656
ChkRange	Проверка на принадлежность заданному диапазону	Определяет, находится ли значение переменной в пределах заданного диапазона значений.	стр. 2-1658
GetMyTaskStatus	Чтение состояния текущей задачи	Считывает состояние текущей задачи.	стр. 2-1661
GetMyTaskInterval	Чтение периода текущей задачи	Служит для чтения периода выполнения текущей задачи.	стр. 2-1664
Task_IsActive	Определение состояния задачи	Позволяет определить, выполняется ли в данный момент указанная задача.	стр. 2-1667
Lock	Блокировка задач	Означает начало эксклюзивной блокировки между задачами. Выполнение любой другой задачи, включающей область блокировки с таким же номером блокировки, становится невозможно.	стр. 2-1669
Unlock	Разблокировка задач	Означает конец эксклюзивной блокировки между задачами.	стр. 2-1669
ActEventTask	Инициация событийной задачи	Инициирует выполнение событийной задачи.	стр. 2-1675
Get**Clk	Группа получения тактовых импульсов	Выдает последовательность тактовых импульсов с заданным периодом.	стр. 2-1682
Get**Cnt	Группа получения значения автономного прямого счетчика	Возвращает текущее значение автономного счетчика заданных интервалов.	стр. 2-1684

2

Описание команд

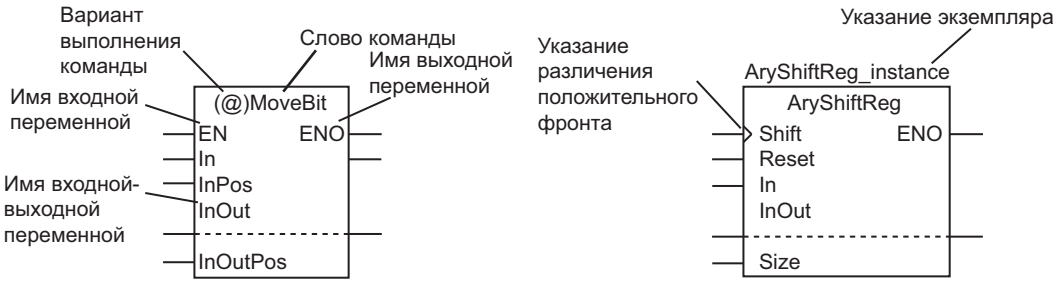
В данном разделе описываются команды, которые могут использоваться со следующим устройством: Серия NJ/NX, модули ЦПУ.

Использование данной главы

Ниже поясняются термины и понятия, которые используются в данной главе для описания команд программирования.

Термины и понятия

Используемые термины и понятия поясняются в таблице ниже.

Термин	Описание
Команда	Приводится слово команды языка программирования. Пример: MoveBit
Имя	Приводится имя команды языка программирования. Пример: Перемещение бита
FB/FUN	Указывает, является ли команда командой функционального блока (FB) или командой функции (FUN). Команды FB можно вызывать только из программ и функциональных блоков. Команды FUN можно вызывать из программ, функциональных блоков и функций.
Графическое представление	Приводится изображение команды, которое используется для представления команды в лестничной диаграмме. ●Пример для команды FUN ●Пример для команды FB  Ниже поясняются понятия *вариант выполнения команды*, *различение положительного фронта* и *указание экземпляра*. Вариант выполнения команды : Возможность указать вариант выполнения команды указывается значком «(@)» перед словом команды FUN. Если для команды указана поддержка вариантов выполнения, перед словом команды можно разместить значок «@», означающий различение положительного фронта. Команда, для которой указано различение положительного фронта, выполняется, только если значение входной переменной EN было ЛОЖЬ в предыдущем цикле выполнения задачи и стало ИСТИНА в текущем цикле выполнения задачи. Различение положительного фронта : Указывается стрелкой, направленной внутрь команды в точке ввода входной переменной. Если для команды указано различение положительного фронта, она различает положительный фронт (перепад) входного сигнала. Указание экземпляра : Экземпляр команды указывается с помощью строки *XX_instance*, размещаемой над командой FB. Любому указанному экземпляру команды должно быть присвоено имя экземпляра.

Термин	Описание
Выражение языка ST	Приводится обозначение команды, которое используется для представления команды в программе на языке структурированного текста (ST). Команду в программе на языке ST можно использовать двумя способами, которые описаны ниже. 1. Прямое указание соответствия между параметрами и входными, выходными и входными-выходными переменными. Пример: MoveBit(In:=abc, InPos:=def, InOut:=ghi, InOutPos:=jkl); 2. Указание только параметров без указания входных, выходных и входных-выходных переменных. Пример: MoveBit(In, InPos, InOut, InOutPos); В данной главе используется способ 2. Для любой команды, которая вводится в программу в виде «XX_instance(имя_переменной)», должно быть назначено имя экземпляра. Пример: TON_instance (In, PT, Q, ET);
Переменные	• Имя Приводятся входные переменные, выходные переменные и входные-выходные переменные. Пример: In1 Однако переменные, используемые многими командами, не указываются на страницах с описанием отдельных команд. Ниже перечислены восемь переменных, которые используются во многих командах. Эти переменные будут описаны ниже в данном разделе. (EN, ENO, Execute, Done, Busy, Error, ErrorID, ErrorIDEx) • Значение Указывается назначение переменной. Пример: Прямой счетчик • Вход/выход Указывается, какая это переменная: входная, выходная или входная-выходная. • Описание Приводится более развернутое описание переменной, указываются любые ограничения на ее использование. • Диапазон допустимых значений Здесь указывается диапазон значений, которые может принимать переменная. *Зависит от типа данных* — указывает, что диапазон допустимых значений переменной зависит от используемого типа данных. Диапазоны допустимых значений для разных типов данных будут приведены позднее в данной главе. • Единица Указывается единица измерения значений, указываемых для переменной. --- указывает, что единица измерения не требуется. Пример: Байты • По умолчанию Если команде перед ее выполнением не назначен параметр, для переменной автоматически используется указанное значение по умолчанию. --- означает следующее: Входные переменные : Входной переменной присваивается значение по умолчанию, соответствующее ее типу данных. Значения по умолчанию для разных типов данных будут приведены позднее в данной главе. Если входная переменная является структурой, значение по умолчанию приводится в характеристиках структуры в описании функции команды. Выходная переменная : Значения по умолчанию не задаются. Входные-выходные переменные : Значения по умолчанию не задаются. • Типы данных Указывается тип данных переменной. Также указывается использование перечислений, массивов, структур и объединений.

Термин	Описание
Функция	Описывается функция, выполняемая командой. Для имен переменных используется курсив. Пример: *In1* После имен массивов следуют квадратные скобки: []. Пример: InOut[]
Связанные системные переменные	Приводятся системные переменные, связанные с командой. Дополнительные сведения о системных переменных см. в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
Связанные переменные, частично определяемые пользователем	Приводятся относящиеся к команде переменные, частично определяемые пользователем, а также их имена. Дополнительные сведения о системных переменных, частично определяемых пользователем, см. в указанных руководствах.
Дополнительная информация	Приводится дополнительная информация о функции, выполняемой командой. Она включает сведения о связанных командах и полезную информацию о применении команды.
Меры предосторожности для обеспечения надлежащей эксплуатации	Приводятся меры предосторожности, которые необходимо соблюдать при использовании команды. Здесь также приводятся условия, при которых могут возникать ошибки при использовании и работе данной команды.
Пример программы	Приводятся короткие примеры использования команды в прикладной программе. Для каждой задачи приводятся примеры на языке лестничных диаграмм и на языке структурированного текста.

Общие переменные

В данном разделе описываются переменные, которые используются для многих команд (EN, ENO, Execute, Done, Busy, Error, ErrorID и ErrorIDEx).
Эти переменные не будут описываться в таблицах переменных для отдельных команд в последующих разделах. Определить, используются ли для той или иной команды эти переменные, можно по графическому представлению или выражению на языке ST для команды.

EN

EN — это входная переменная, которая выполняет роль условия выполнения для команды FUN. При использовании команды FUN в лестничной диаграмме условие выполнения следует подключить ко входу *EN*.

	Значение	Вход/выход	Описание	Тип данных	Диапазон допустимых значений	По умолчанию
EN	Активация (условие выполнения)	Вход	ИСТИНА: команда выполняется. *1 ЛОЖЬ: команда не выполняется.	BOOL	ИСТИНА или ЛОЖЬ	ИСТИНА

*1. Если для команды указан вариант выполнения с различением положительного фронта (@), команда выполняется, когда значение *EN* меняется с ЛОЖЬ на ИСТИНА.

- У команд FB входная переменная *EN* отсутствует.
- При вызове команды FUN из структурированного текста входную переменную *EN* следует опускать. В структурированном тексте нет необходимости в использовании входной переменной *EN*, поскольку условие выполнения команды определяется последовательностью операций.

ENO

ENO — это выходная переменная, которая передает результат выполнения команды.
В лестничной диаграмме выходная переменная может использоваться для передачи условий выполнения следующей команде.
При завершении выполнения команды значение переменной *ENO* обычно меняется на ИСТИНА. После этого начинается выполнение следующей команды.

	Значение	Вход/выход	Описание	Тип данных	Диапазон допустимых значений	По умолчанию
ENO	Выход активации	Выход	ИСТИНА: нормальное завершение.*1 ЛОЖЬ: завершение с ошибкой, идет выполнение или условие выполнения не удовлетворяется.	BOOL	ИСТИНА или ЛОЖЬ	---

*1. Переменная содержит значение ИСТИНА, только пока удовлетворяется условие выполнения. Когда после нормального завершения команды условие выполнения перестает удовлетворяться, переменная *ENO* принимает значение ЛОЖЬ.

- Выходную переменную *ENO* имеют большинство команд FUN и FB. У некоторых команд, впрочем, выходная переменная *ENO* отсутствует.

Execute, Done и Busy

Execute — это входная переменная, которая выполняет роль условия выполнения для некоторых команд FB.
Выполнение команды запускается, когда значение *Execute* меняется на ИСТИНА. После этого выполнение команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи.

Done — это выходная переменная в некоторых командах FB, которая показывает, что выполнение команды завершено.
Busy — это выходная переменная в некоторых командах FB, которая показывает, что команда выполняется в данный момент.

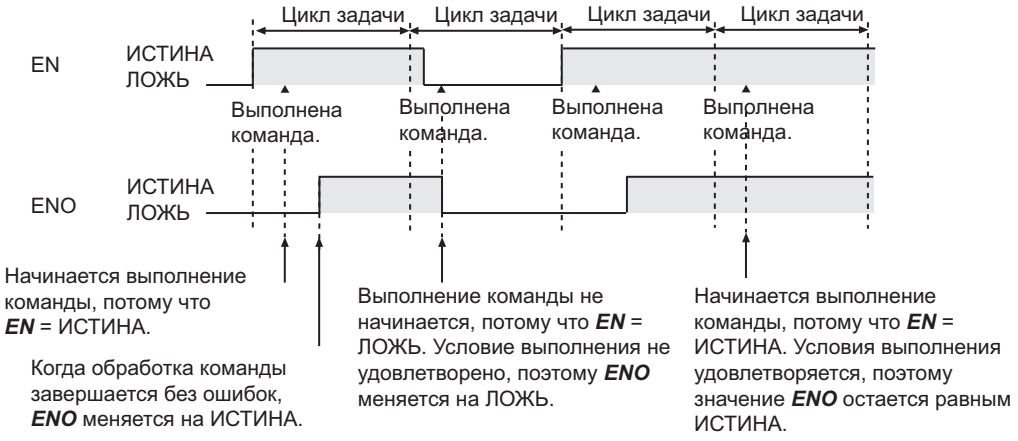
	Значение	Вход/выход	Описание	Тип данных	Диапазон допустимых значений	По умолчанию
Execute	Выполнить	Вход	ИСТИНА: команда выполняется. *1 ЛОЖЬ: команда не выполняется. *2	BOOL	ИСТИНА или ЛОЖЬ	ЛОЖЬ

	Значение	Вход/ выход	Описание	Тип данных	Диапазон допустимых значений	По умолчанию
Done	Завершено	Выход	ИСТИНА: нормальное завершение.*3 *4 ЛОЖЬ: завершение с ошибкой, идет выполнение или условие выполнения не удовлетворяется.	BOOL	ИСТИНА или ЛОЖЬ	---
Busy	Занято		ИСТИНА: выполняется в данный момент. ЛОЖЬ: не выполняется в данный момент.			

*1. Если переменная *Execute* уже содержит значение ИСТИНА на момент начала работы контроллера, команда не выполняется. Чтобы команда была выполнена, следует поменять значение *Execute* на ЛОЖЬ.
*2. Обработка продолжается до конца, даже если значение этой переменной меняется на ЛОЖЬ во время выполнения.
*3. Когда после нормального завершения команды условие выполнения перестает удовлетворяться, переменная *Done* принимает значение ЛОЖЬ.
*4. Если условие выполнения уже не удовлетворяется, когда команда нормально завершается, *Done* остается в состоянии ИСТИНА в течение одного цикла выполнения задачи, а затем переходит в состояние ЛОЖЬ.

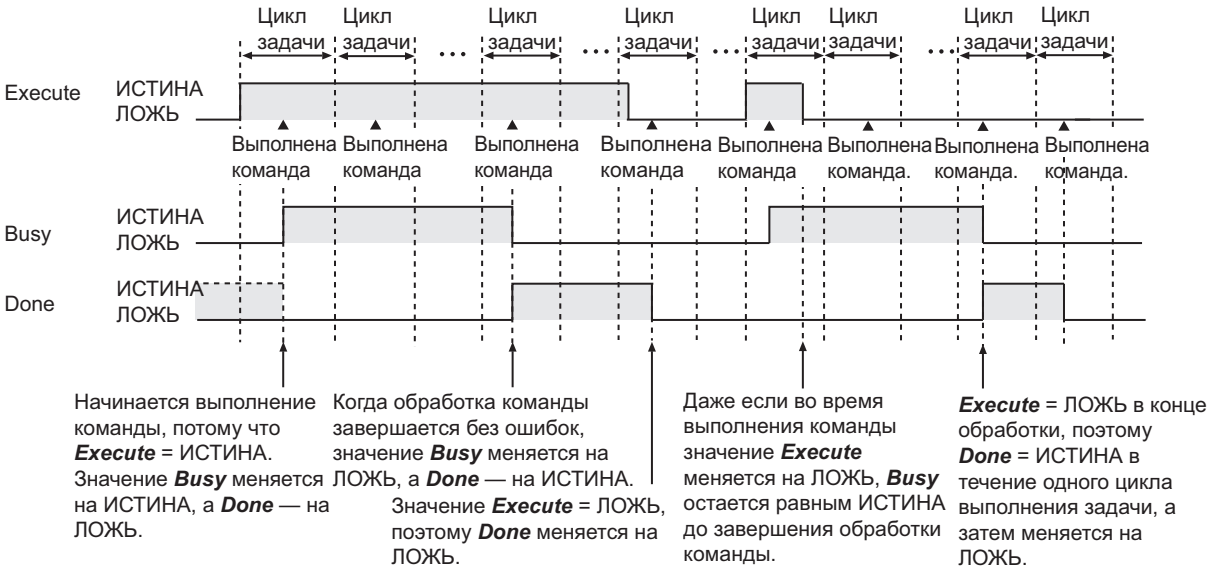
● Команды, выполняемые за один период выполнения задачи

Ниже приведена временная диаграмма для команды, имеющей переменные *EN* и *ENO* (т. е. команды, которая должна завершиться за один цикл выполнения задачи).



● Команды, обрабатываемые дольше одного цикла выполнения задачи

Ниже приведена временная диаграмма для команды, имеющей переменные *Execute* и *Busy* (т. е. команды, которая обрабатывается дольше одного цикла выполнения задачи).



Error, ErrorID и ErrorIDEx

Error, *ErrorID* и *ErrorIDEx* — это выходные переменные, которые используются в некоторых командах FB для индикации завершения команды с ошибкой

	Значение	Вход/ выход	Описание	Тип данных	Диапазон допустимых значений	По умолчанию
Error	Ошибка	Выход	ИСТИНА: завершение с ошибкой.*1 *2 ЛОЖЬ: нормальное завершение, идет выполнение или условие выполнения не удовлетворяется.	BOOL	ИСТИНА или ЛОЖЬ	---
ErrorID	Код ошибки		Идентификатор ошибки при завершении команды с ошибкой При нормальном завершении переменная содержит значение WORD#16#0.	WORD	Зависит от команды.	
ErrorIDEx	Дополнительный код ошибки		Это идентификатор ошибки в случае аппаратной ошибки модуля расширения. При нормальном завершении переменная содержит значение DWORD#16#0.	DWORD		

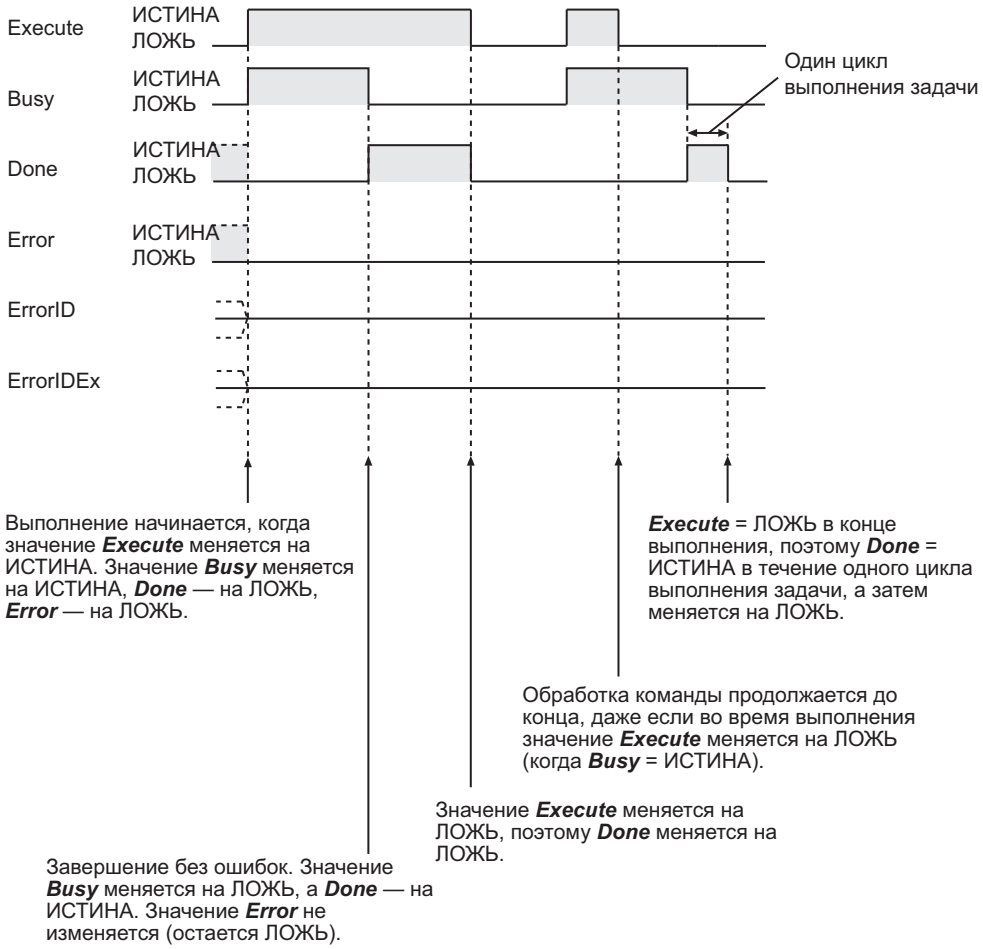
*1. Когда после завершения команды с ошибкой условие выполнения перестает удовлетворяться, значение переменной *Error* меняется на ЛОЖЬ.

*2. Если условие выполнения уже не удовлетворяется, когда команда завершается с ошибкой, выход *Error* остается в состоянии ИСТИНА в течение одного цикла выполнения задачи, а затем переходит в состояние ЛОЖЬ.

См. информацию в списке кодов ошибок в *A-1 Коды ошибок в переменной ErrorID* на стр. A-2 для соответствующего идентификатора *ErrorID*.
Значения кодов ошибок см. в документе *Серия NJ/NX — Поиск и устранение неполадок. Руководство (Cat. No. W503)*.

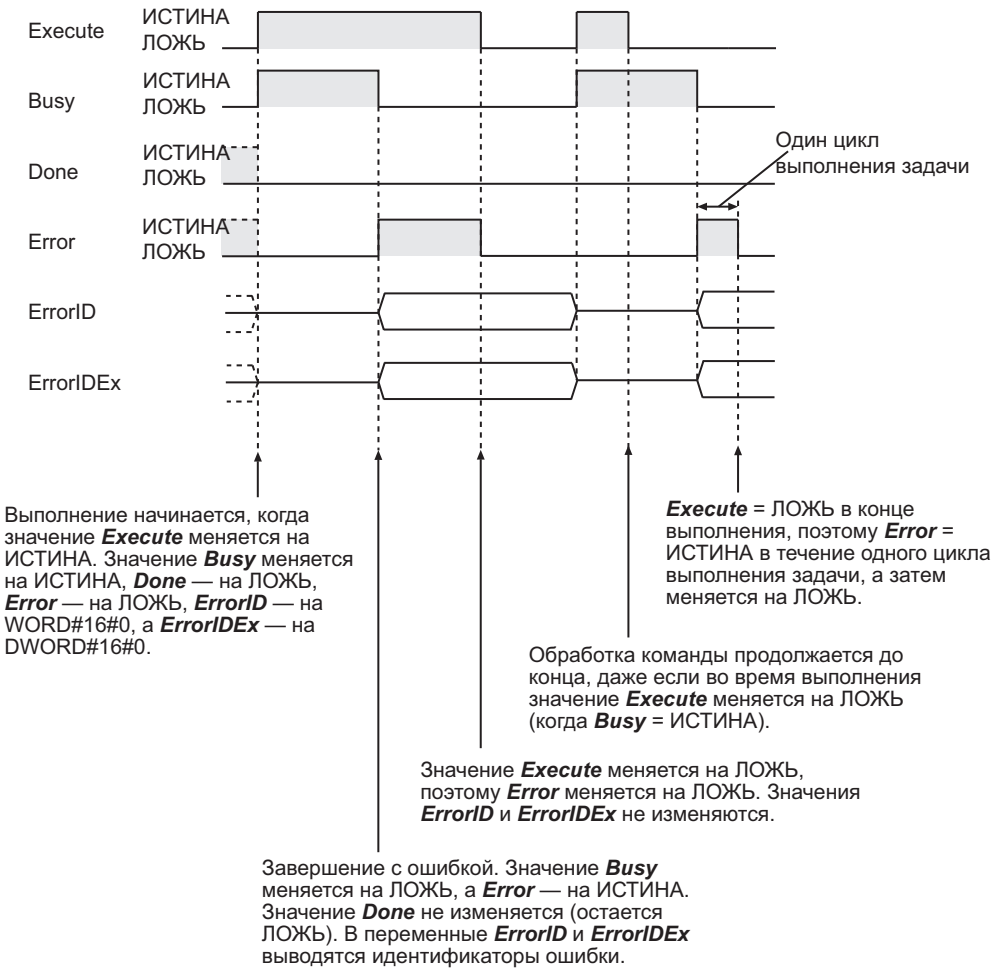
● Нормальное завершение

Ниже приведена временная диаграмма, демонстрирующая работу переменных *Execute*, *Done*, *Busy*, *Error*, *ErrorID* и *ErrorIDEx*.



● Завершение с ошибкой

Ниже приведена временная диаграмма, демонстрирующая работу переменных *Execute*, *Done*, *Busy*, *Error*, *ErrorID* и *ErrorIDEx*.



Диапазоны допустимых значений и значения переменных по умолчанию

Диапазон допустимых значений переменной — это диапазон значений, которые может принимать переменная. Значение переменной по умолчанию — это значение, которое присваивается входной переменной, когда команда выполняется без назначения параметра входной переменной.

Эти значения определяются для каждого типа данных. Если для команды не указаны конкретные значения, применяются диапазоны допустимых значений и значения по умолчанию, соответствующие типам данных переменных.

Для таких переменных в столбце диапазона допустимых значений указывается *Зависит от типа данных*, а в столбце значения по умолчанию для входных переменных указывается ---.

В следующих таблицах приведены диапазоны допустимых значений и значения по умолчанию для разных типов данных.

Классификация	Тип данных	Диапазон допустимых значений	По умолчанию
Логический тип	BOOL	ИСТИНА или ЛОЖЬ	ЛОЖЬ

Классификация	Тип данных	Диапазон допустимых значений	По умолчанию
Битовая строка	BYTE	BYTE#16#00...FF	BYTE#16#00
	WORD	WORD#16#0000...FFFF	WORD#16#0000
	DWORD	DWORD#16#00000000...FFFFFFFF	DWORD#16#00000000
	LWORD	LWORD#16#0000000000000000...FFFFFFFFFFFFFFFF	LWORD#16#0000000000000000
Целочисленные типы	USINT	USINT#0...+255	USINT#0
	UINT	UINT#0...+65535	UINT#0
	UDINT	UDINT#0...+4294967295	UDINT#0
	ULINT	ULINT#0...+18446744073709551615	ULINT#0
	SINT	SINT#-128...+127	SINT#0
	INT	INT#-32768...+32767	INT#0
	DINT	DINT#-2147483648...+2147483647	DINT#0
	LINT	LINT#-9223372036854775808...+9223372036854775807	LINT#0
Вещественные типы	REAL	REAL#-3,402823e+38...-1,175495e-38, 0, +1,175494e-38...+3,402823e+38, +∞/-∞	REAL#0
	LREAL	LREAL#-1,79769313486231e+308...-2,22507385850721e-308, 0, +2,22507385850721e-308...+1,79769313486231e+308, +∞/-∞	LREAL#0
Значения времени и продолжительности, даты и текстовые строки	TIME	T#-9223372036854,775808ms (T#-106751d_23h_47m_16s_854,775808ms)...T#9223372036854,775807ms (T#+106751d_23h_47m_16s_854,775807ms)	T#0s
	DATE	D#1970-01-01...D#2106-02-06 (1 января 1970 года...6 февраля 2106 года)	D#1970-01-01
	TOD	TOD#00:00:00,000000000...TOD#23:59:59,999999999 (00:00 и 0,000000000...23:59 и 59,999999999 с)	TOD#00:00:00,000000000
	DT	DT#1970-01-01-00:00:00,000000000...DT#2106-02-06-23:59:59,999999999 (00:00 и 0,000000000, 1 января 1970 года...23:59 и 59,999999999 с, 6 февраля 2106 года)	DT#1970-01-01-00:00:00.000000000
	STRING	Кодировка: UTF-8 0...1 986 байт (1 985 однобайтовых буквенно-цифровых символов + последний символ NULL)	''

Производные типы данных (перечисления, структуры и объединения)

Для переменных, относящихся к производным типам данных (перечисления, структуры и объединения), в таблицах типов данных переменных указываются соответствующие производные типы данных. Ниже приводятся примеры для каждого типа данных.

Перечисления

В таблице указывается тип данных для переменной перечислимого типа.
Пример приведен ниже. В данном случае для переменной *Out* используется перечислимый тип данных _eDAYOFWEEK. Перечислители описываются в описании функции команды.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																	OK		OK	
Out	Сведения о перечислителях перечислимого типа _eDAYOFWEEK см. в разделе «Функция».																			

Структуры и объединения

В таблице указывается тип данных для переменной типа «структура» или «объединение».
Пример приведен ниже. В данном случае для переменной *In1* используется структурный тип данных _sPORT. Сведения о членах структуры или объединения приводятся в описании функции команды.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	Подробные сведения о структуре _sPORT см. в разделе «Функция».																			

В таблице также указываются любые переменные, для которых в качестве параметра может быть указана структура, член структуры, объединение или член объединения.
В приведенном ниже примере для переменной *In1* можно указать параметр базового типа данных или параметр, являющийся структурой, членом структуры, объединением или членом объединения. Если нужно использовать переменную типа «структура» или «объединение», в качестве параметра нужно указать именно структуру или объединение. Если переменная должна быть членом структуры или объединения, следует указать член структуры или объединения в качестве параметра.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать структуру, член структуры, объединение или член объединения.																			

Указание переменных-массивов

Имена переменных, являющихся массивами, заканчиваются квадратными скобками «[]» и строкой «(array)». Для таких переменных в качестве параметра следует указывать элемент массива (т. е. должен указываться индекс элемента массива).
Пример приведен ниже. В данном случае в таблице указывается, что переменная In1[] представляет собой массив с элементами типа BYTE.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)		OK																		

В таблице типов данных указываются массивы, в которых в качестве элементов могут использоваться структуры и объединения. См. пример ниже. Для таких переменных в качестве параметра следует указывать элемент массива (т. е. должен указываться индекс элемента массива).

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)	Также допускается указывать массивы структур или объединений.																			

В таблице указываются любые переменные, для которых в качестве параметра можно указать массив или элемент массива
В примере ниже для переменной *In1* можно указать базовый тип данных либо можно указать весь массив целиком или элемент массива. Для указания массива целиком в качестве параметра передается имя массива. Для указания элемента массива в качестве параметра следует передать имя массива с указанием индекса элемента.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать массив или элемент массива.																			

Прочее

Ошибки, обнаруживаемые для всех команд

Ошибки, которые могут возникать при использовании команды, приводятся в разделе «Меры предосторожности для обеспечения надлежащей эксплуатации» в описании данной команды. Однако есть ряд ошибок, которые могут возникать в связи с любой командой. Эти ошибки не перечисляются в разделах «Меры предосторожности для обеспечения надлежащей эксплуатации». Это следующие ошибки:

- Чтение или запись элементов вне диапазона переменной массива.
  Пример: Указание входной переменной a[4] для переменной-массива a[0..3].
- Передача параметров, не являющихся переменными-массивами, в команды, входные, выходные или входные-выходные переменные которых определены как переменные массивы.
- Присвоение переменной типа STRING текстовой строки, длина которой превышает установленное количество байтов.
- Присвоение переменной типа STRING текстовой строки, которая не завершается нулевым символом (NULL).
- Деление целочисленной переменной на 0.

Меры предосторожности для всех команд

Объем обработки, требуемый для выполнения некоторых команд, зависит от параметров, которые подключены к команде.
Если объем обработки велик, время выполнения команды возрастает и может превысить продолжительность цикла выполнения задачи. Это приводит к ошибке «Превышение времени цикла задачи». В этом случае следует уменьшить объем обработки до приемлемого.

Команды языка релейно-контактных схем

Команда	Имя	Стр.
LD и LDN	ЗАГРУЗКА и ЗАГРУЗКА НЕ	стр. 2-16
AND и ANDN	И/И НЕ	стр. 2-19
OR и ORN	ИЛИ/ИЛИ НЕ	стр. 2-22
Out и OutNot	ВЫВОД и ВЫВОД НЕ	стр. 2-25

LD и LDN

LD : Считывает значение переменной типа BOOL.
LDN : Считывает инвертированное значение переменной типа BOOL.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LD	Загрузка	---		Нет
LDN	Загрузка НЕ	---		Нет

Переменные

Нет

Функция

LD

Команда LD считывает значение указанной переменной типа BOOL и выводит его в следующую команду.
Если указанная переменная содержит значение «ИСТИНА», то выводится «ИСТИНА». Если значение равно «ЛОЖЬ», выводится «ЛОЖЬ».
Команду LD используют в качестве самого первого нормально разомкнутого контакта (бита) логической цепи (присоединяемого непосредственно к левой шине) или самого первого нормально разомкнутого контакта (бита) логического блока.

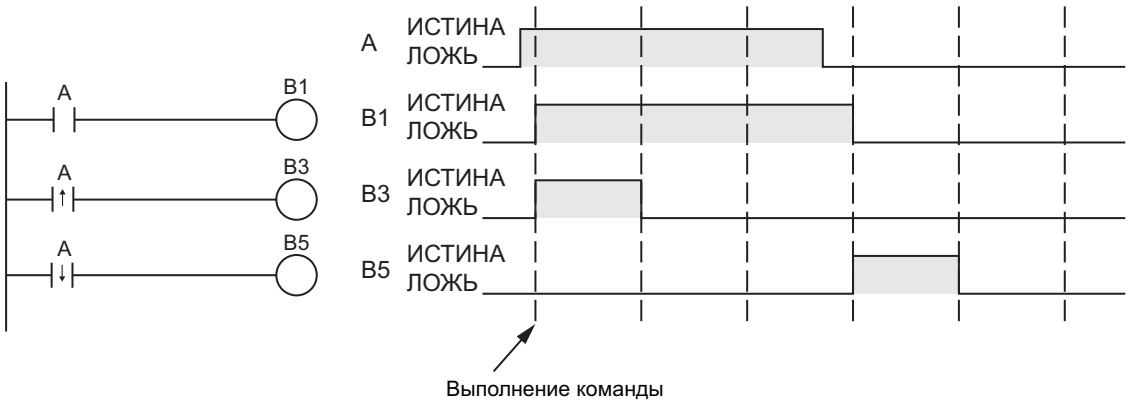
Ниже показана работа команды для случая, когда не указано выделение положительного или отрицательного фронта.

Команда	Значение переменной	Выходное значение
LD	ИСТИНА	ИСТИНА
	ЛОЖЬ	ЛОЖЬ

Если указано различение положительного или отрицательного фронта, выполнение команды зависит от следующих факторов: значение переменной во время последнего выполнения команды и текущее значение переменной. Сказанное описано в таблице ниже.

Команда	Различение фронта	Значение переменной при последнем выполнении и текущее значение переменной	Выходное значение
LD	Различение положительного фронта	ЛОЖЬ при последнем выполнении → сейчас ИСТИНА	ИСТИНА
		Прочее, кроме вышеуказанного.	ЛОЖЬ
	Различение отрицательного фронта	ИСТИНА при последнем выполнении → сейчас ЛОЖЬ	ИСТИНА
		Прочее, кроме вышеуказанного.	ЛОЖЬ

Пример программы и временная диаграмма показаны на рисунке ниже.



LDN

Команда LDN считывает значение указанной переменной типа BOOL, инвертирует его и выводит результат в следующую команду.
Если указанная переменная содержит значение «ИСТИНА», то выводится «ЛОЖЬ». Если значение равно «ЛОЖЬ», выводится «ИСТИНА».
Команду LDN используют в качестве самого первого нормально замкнутого контакта (бита) логической цепи (присоединяемого непосредственно к левой шине) или самого первого нормально замкнутого контакта (бита) логического блока.

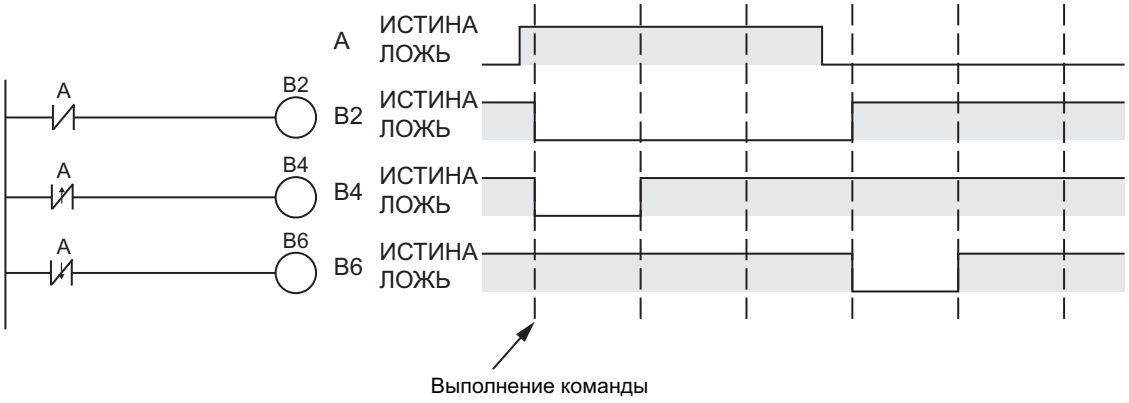
Ниже показана работа команды для случая, когда не указано выделение положительного или отрицательного фронта.

Команда	Значение переменной	Выходное значение
LDN	ИСТИНА	ЛОЖЬ
	ЛОЖЬ	ИСТИНА

Если указано различение положительного или отрицательного фронта, выполнение команды зависит от следующих факторов: значение переменной во время последнего выполнения команды и текущее значение переменной. Сказанное описано в таблице ниже.

Команда	Различение фронта	Значение переменной при последнем выполнении и текущее значение переменной	Выходное значение
LDN	Различение положительного фронта	ЛОЖЬ при последнем выполнении → сейчас ИСТИНА	ЛОЖЬ
		Прочее, кроме вышеуказанного.	ИСТИНА
	Различение отрицательного фронта	ИСТИНА при последнем выполнении → сейчас ЛОЖЬ	ЛОЖЬ
		Прочее, кроме вышеуказанного.	ИСТИНА

Пример программы и временная диаграмма показаны на рисунке ниже.



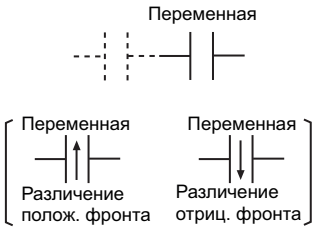
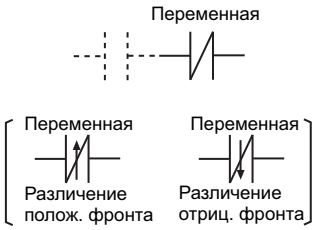
Меры предосторожности для обеспечения надлежащей эксплуатации

- В указанном ниже случае возникает ошибка и сохраняется выходное значение, полученное при последнем выполнении.
  a) Для значения переменной указан элемент массива, которого не существует.
  Пример: Определен массив a[0..5] типа BOOL, но при выполнении команды в качестве переменной используется a[10].
- Не используйте эти команды в качестве самой правой команды в цепи. В противном случае в Sysmac Studio возникнет ошибка и загрузить программу пользователя в контроллер будет невозможно.

AND и ANDN

AND : Выполняет операцию «логическое И» над значением переменной типа BOOL и входным значением.

ANDN : Выполняет операцию «логическое И» над инвертированным значением переменной типа BOOL и входным значением.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AND	И	---		result:=vBool1 AND vBool2; result:=vBool1 & vBool2;
ANDN	И НЕ	---		result:=vBool1 AND NOT vBool2;

Переменные

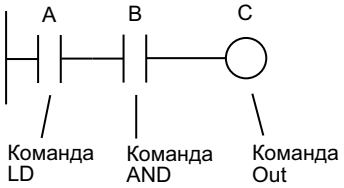
Нет

Функция

AND

Команда AND выполняет операцию «логическое И» над значением указанной переменной типа BOOL и условием выполнения и выводит результат в следующую команду.
Команду AND используют в качестве нормально разомкнутого контакта (бита), соединяемого последовательно с предыдущей командой.

Пример программы с использованием команды AND показан на рисунке ниже. Она выполняет операцию «логическое И» над значениями переменных *A* и *B* и выводит результат в переменную *C*.



Выполняет операцию «логическое И» над переменными *A* и *B* и выводит результат в переменную *C*.

Ниже показана работа команды для случая, когда не указано выделение положительного или отрицательного фронта.

Команда	Комбинация значения переменной и условия выполнения	Выходное значение
AND	Значение переменной: ИСТИНА Условие выполнения: ИСТИНА	ИСТИНА
	Прочее, кроме вышеуказанного.	ЛОЖЬ

Если указано различение положительного или отрицательного фронта, выполнение команды зависит от следующих факторов: значение переменной во время последнего выполнения команды, текущее значение переменной и условие выполнения. Сказанное описано в таблице ниже.

Команда	Различение фронта	Комбинация значения переменной при последнем выполнении, текущего значения переменной и условия выполнения	Выходное значение
AND	Различение положительного фронта	Значение переменной: ЛОЖЬ при последнем выполнении → сейчас ИСТИНА Условие выполнения: ИСТИНА	ИСТИНА
		Прочее, кроме вышеуказанного.	ЛОЖЬ
	Различение отрицательного фронта	Значение переменной: ИСТИНА при последнем выполнении → сейчас ЛОЖЬ Условие выполнения: ИСТИНА	ИСТИНА
		Прочее, кроме вышеуказанного.	ЛОЖЬ

ANDN

Команда ANDN выполняет операцию «логическое И» над инвертированным значением указанной переменной типа BOOL и условием выполнения и выводит результат в следующую команду. Команду ANDN используют в качестве нормально замкнутого контакта (бита), соединяемого последовательно с предыдущей командой.

Ниже показана работа команды для случая, когда не указано выделение положительного или отрицательного фронта.

Команда	Комбинация значения переменной и условия выполнения	Выходное значение
ANDN	Значение переменной: ЛОЖЬ Условие выполнения: ИСТИНА	ИСТИНА
	Прочее, кроме вышеуказанного.	ЛОЖЬ

Если указано различение положительного или отрицательного фронта, выполнение команды зависит от следующих факторов: значение переменной во время последнего выполнения команды, текущее значение переменной и условие выполнения. Сказанное описано в таблице ниже.

Команда	Различение фронта	Комбинация значения переменной при последнем выполнении, текущего значения переменной и условия выполнения	Выходное значение
ANDN	Различение положительного фронта	Значение переменной: ЛОЖЬ при последнем выполнении → сейчас ИСТИНА Условие выполнения: ИСТИНА	ЛОЖЬ
		Значение переменной: Игнорируется Условие выполнения: ЛОЖЬ	
		Прочее, кроме вышеуказанного.	ИСТИНА
	Различение отрицательного фронта	Значение переменной: ИСТИНА при последнем выполнении → сейчас ЛОЖЬ Условие выполнения: ИСТИНА	ЛОЖЬ
		Значение переменной: Игнорируется Условие выполнения: ЛОЖЬ	
		Прочее, кроме вышеуказанного.	ИСТИНА

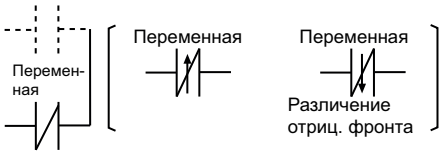
Меры предосторожности для обеспечения надлежащей эксплуатации

- В указанном ниже случае возникает ошибка и сохраняется выходное значение, полученное при последнем выполнении.
  a) Для значения переменной указан элемент массива, которого не существует.
  Пример: Определен массив a[0..5] типа BOOL, но при выполнении команды в качестве переменной используется a[10].
- Не используйте эти команды в качестве самой правой команды в цепи. В противном случае в Sysmac Studio возникнет ошибка и загрузить программу пользователя в контроллер будет невозможно.
- Эти команды невозможно подключать непосредственно к шине.

OR и ORN

OR : Выполняет операцию «логическое ИЛИ» над значением переменной типа BOOL и условием выполнения.

ORN : Выполняет операцию «логическое ИЛИ» над инвертированным значением переменной типа BOOL и условием выполнения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
OR	ИЛИ	---		result:=vBool1 OR vBool2;
ORN	ИЛИ НЕ	---		result:=vBool1 OR NOT vBool2;

Переменные

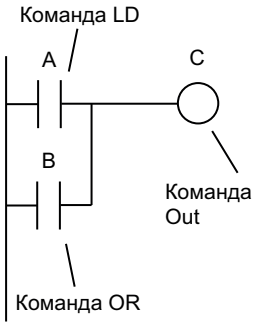
Нет

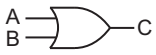
Функция

OR

Команда OR выполняет операцию «логическое ИЛИ» над значением указанной переменной типа BOOL и условием выполнения и выводит результат в следующую команду.
Команду OR используют в качестве нормально разомкнутого контакта (бита), соединяемого параллельно предыдущей команде. Команда OR служит для выполнения операции «логическое ИЛИ» над нормально разомкнутым контактом (битом) и одним из следующих операндов: командой LD или LDN, подключенной непосредственно к шине, либо логическим блоком, начинающимся с команды LD или LDN и завершающимся командой, которая непосредственно предшествует команде OR.

Пример программы с использованием команды OR показан на рисунке ниже. Она выполняет операцию «логическое ИЛИ» над значениями переменных *A* и *B* и выводит результат в переменную *C*.





Выполняет операцию «логическое ИЛИ» над переменными *A* и *B* и выводит результат в переменную *C*.

Ниже показана работа команды для случая, когда не указано выделение положительного или отрицательного фронта.

Команда	Комбинация значения переменной и условия выполнения	Выходное значение
OR	Значение переменной: ЛОЖЬ Условие выполнения: ЛОЖЬ	ЛОЖЬ
	Прочее, кроме вышеуказанного.	ИСТИНА

Если указано различение положительного или отрицательного фронта, выполнение команды зависит от следующих факторов: значение переменной во время последнего выполнения команды, текущее значение переменной и условие выполнения. Сказанное описано в таблице ниже.

Команда	Различение фронта	Комбинация значения переменной при последнем выполнении, текущего значения переменной и условия выполнения	Выходное значение
OR	Различение положительного фронта	Значение переменной: ЛОЖЬ при последнем выполнении → сейчас ИСТИНА Условие выполнения: игнорируется.	ИСТИНА
		Значение переменной: игнорируется. Условие выполнения: ИСТИНА	
		Прочее, кроме вышеуказанного.	ЛОЖЬ
	Различение отрицательного фронта	Значение переменной: ИСТИНА при последнем выполнении → сейчас ЛОЖЬ Условие выполнения: игнорируется.	ИСТИНА
		Значение переменной: игнорируется. Условие выполнения: ИСТИНА	
		Прочее, кроме вышеуказанного.	ЛОЖЬ

ORN

Команда ORN выполняет операцию «логическое ИЛИ» над инвертированным значением указанной переменной типа BOOL и условием выполнения и выводит результат в следующую команду.

Команду ORN используют в качестве нормально замкнутого контакта (бита), соединяемого параллельно предыдущей команде. Команда ORN служит для выполнения операции «логическое ИЛИ» над нормально замкнутым контактом (битом) и одним из следующих операндов: командой LD или LDN, подключенной непосредственно к шине, либо логическим блоком, начинающимся с

команды LD или LDN и завершающимся командой, которая непосредственно предшествует команде ORN.

Ниже показана работа команды для случая, когда не указано выделение положительного или отрицательного фронта.

Команда	Комбинация значения переменной и условия выполнения	Выходное значение
ORN	Значение переменной: ИСТИНА Условие выполнения: ЛОЖЬ	ЛОЖЬ
	Прочее, кроме вышеуказанного.	ИСТИНА

Если указано различение положительного или отрицательного фронта, выполнение команды зависит от следующих факторов: значение переменной во время последнего выполнения команды, текущее значение переменной и условие выполнения. Сказанное описано в таблице ниже.

Команда	Различение фронта	Комбинация значения переменной при последнем выполнении, текущего значения переменной и условия выполнения	Выходное значение
ORN	Различение положительного фронта	Значение переменной: ЛОЖЬ при последнем выполнении → сейчас ИСТИНА Условие выполнения: ЛОЖЬ	ЛОЖЬ
		Прочее, кроме вышеуказанного.	ИСТИНА
	Различение отрицательного фронта	Значение переменной: ИСТИНА при последнем выполнении → сейчас ЛОЖЬ Условие выполнения: ЛОЖЬ	ЛОЖЬ
		Прочее, кроме вышеуказанного.	ИСТИНА

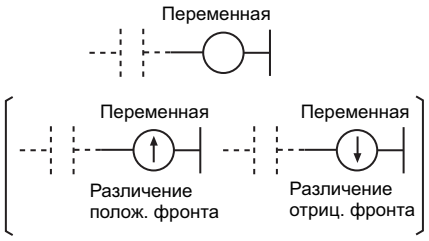
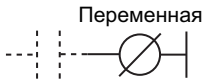
Меры предосторожности для обеспечения надлежащей эксплуатации

- В указанном ниже случае возникает ошибка и сохраняется выходное значение, полученное при последнем выполнении.
  a) Для значения переменной указан элемент массива, которого не существует.
  Пример: Определен массив a[0..5] типа BOOL, но при выполнении команды в качестве переменной используется a[10].
- Не используйте эти команды в качестве самой правой команды в цепи. В противном случае в Sysmac Studio возникнет ошибка и загрузить программу пользователя в контроллер будет невозможно.

Out и OutNot

Out : Берет логический результат выполнения предыдущей команды и выводит его в переменную типа BOOL.

OutNot : Берет инвертированное значение логического результата выполнения предыдущей команды и выводит его в переменную типа BOOL.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Out	Выход	---		Переменная:=(логическое выражение вплоть до предыдущей команды);
OutNot	Выход НЕ	---		Переменная:=NOT(логическое выражение вплоть до предыдущей команды);

Переменные

Нет

Функция

Out

Команда Out берет логический результат выполнения предыдущей команды и выводит его в указанную переменную типа BOOL.

Ниже показана работа команды для случая, когда не указано выделение положительного или отрицательного фронта.

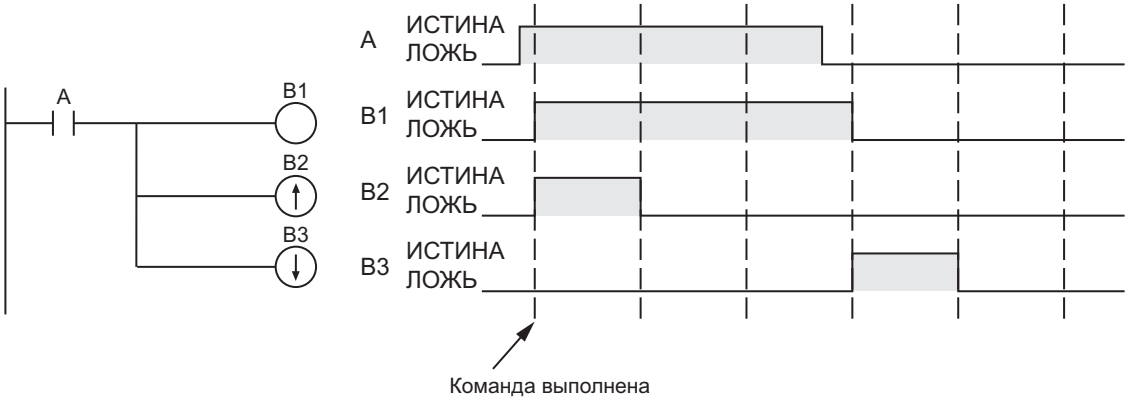
Логический результат выполнения предыдущей команды	Выход
ИСТИНА	ИСТИНА
ЛОЖЬ	ЛОЖЬ

Для команды Out можно указать выполнение по положительному или отрицательному фронту. В том случае, когда указано выделение положительного или отрицательного фронта, выходное значение определяется изменением логического результата выполнения предыдущей команды в период между последним выполнением команды и текущим ее выполнением. Работа команды определяется текущим логическим результатом выполнения предыдущей команды, что отражено в следующей таблице.

Различение фронта	Логические результаты выполнения при предыдущем и текущем выполнении	Выход
Различение положительного фронта	ЛОЖЬ при последнем выполнении → сейчас ИСТИНА	ИСТИНА
	Прочее, кроме вышеуказанного.	ЛОЖЬ

Различение фронта	Логические результаты выполнения при предыдущем и текущем выполнении	Выход
Различение отрицательного фронта	ИСТИНА при последнем выполнении → сейчас ЛОЖЬ	ИСТИНА
	Прочее, кроме вышеуказанного.	ЛОЖЬ

Пример программы и временная диаграмма показаны на рисунке ниже.

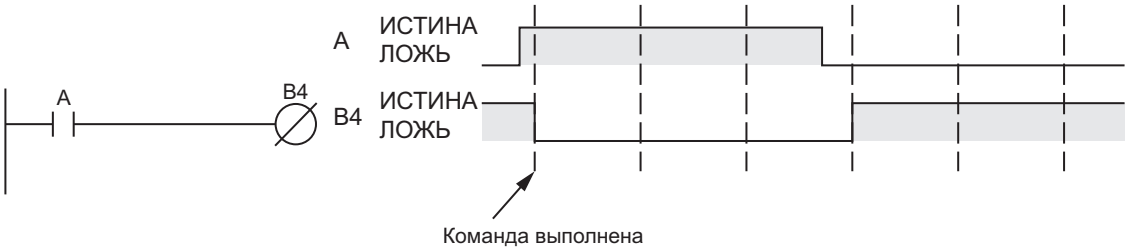


OutNot

Команда OutNot берет инвертированный логический результат выполнения предыдущей команды и выводит его в указанную переменную типа BOOL.

Логический результат выполнения предыдущей команды	Выход
ИСТИНА	ЛОЖЬ
ЛОЖЬ	ИСТИНА

Пример программы и временная диаграмма показаны на рисунке ниже.



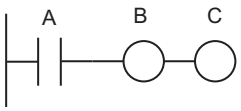
Дополнительная информация

Отличия между командами Set и Reset и командами Out и OutNot

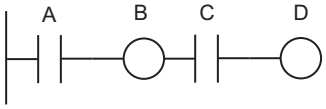
- Команды Set и Reset срабатывают только тогда, когда входное значение меняется на ИСТИНА. При входном значении ЛОЖЬ они не выполняются. Другими словами, при входном значении ЛОЖЬ состояние выхода команды не изменяется.
- Выходное состояние команд Out и OutNot может изменяться при любом логическом результате выполнения предыдущей команды: ИСТИНА или ЛОЖЬ.

Меры предосторожности для обеспечения надлежащей эксплуатации

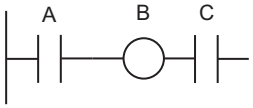
- В указанном ниже случае возникает ошибка и ничего не выводится.
  a) Для значения переменной указан элемент массива, которого не существует.
  Пример: Определен массив a[0..5] типа BOOL, но при выполнении команды в качестве переменной используется a[10].
- Допускаются указанные ниже соединения.
  a) К выходу одной команды Out можно подключить другую команду Out.



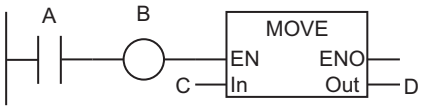
  b) После команды Out можно подключить команду LD и команду Out.



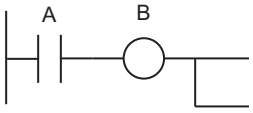
- Указанные ниже соединения не допускаются.
  a) После команды Out нельзя подключить только команду LD.

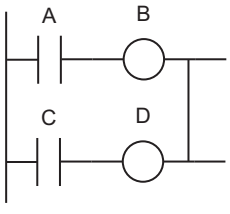


  b) После команды Out нельзя подключать функции и функциональные блоки.



  c) После команд Out не допускается использовать ветвления и присоединения.





Команды для инструкций языка ST

Команда	Имя	Стр.
IF	Если	стр. 2-30
CASE	Case	стр. 2-34
WHILE	Цикл While	стр. 2-38
REPEAT	Цикл Repeat	стр. 2-41
EXIT	Выход из цикла	стр. 2-44
RETURN	Возврат	стр. 2-47
FOR	Начало цикла	стр. 2-48

IF

Конструкция IF выбирает для выполнения одну из двух инструкций на основе результата оценки указанного выражения условия.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST

IF	Если	---	Нет	IF выражение условия THEN
				инструкция;
				ELSIF выражение условия THEN
				инструкция;
				ELSE
				инструкция;
				END_IF;

Переменные

Нет

Функция

Конструкция IF выбирает для выполнения одну из двух инструкций на основе результата оценки указанного выражения условия. Используемое выражение условия должно возвращать состояние ИСТИНА или ЛОЖЬ, как показано в таблице ниже.

Элемент, используемый для выражения условия	Пример	Результат оценки
Логическое выражение	a > 3	Если значение переменной *a* больше 3, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
	a = b	Если значения переменных *a* и *b* равны, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
Переменная типа BOOL	abc	Если значение переменной *abc* = ИСТИНА, то результат = ИСТИНА. Если значение = ЛОЖЬ, то результат = ЛОЖЬ.
Константа типа BOOL	ИСТИНА	ИСТИНА
Функция с возвращаемым значением типа BOOL	Имя функции	Если функция возвращает ИСТИНА, то результат = ИСТИНА. Если она возвращает ЛОЖЬ, то результат = ЛОЖЬ.

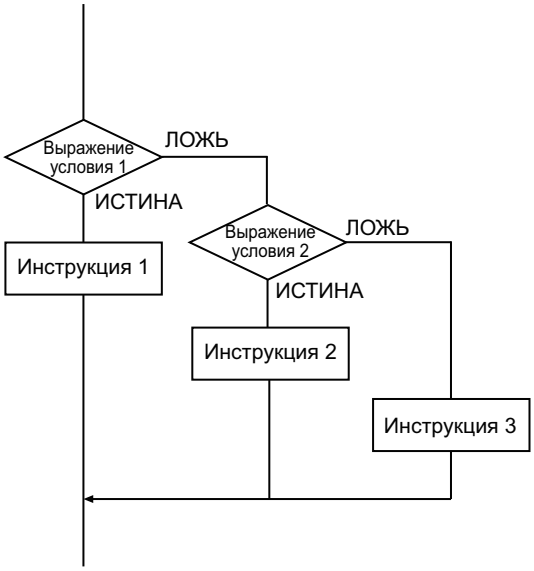
В логическом выражении можно использовать перечисленные ниже операторы.

Оператор	Назначение	Пример	Результат оценки
=	Равно	a = b	Если значения переменных *a* и *b* равны, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
<>	Не равно	a <> b	Если значения переменных *a* и *b* не равны, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
<	Сравнение	a < b	Если значение переменной *a* меньше значения переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
<=		a <= b	Если значение переменной *a* меньше или равно значению переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
>		a > b	Если значение переменной *a* больше значения переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
>=		a >= b	Если значение переменной *a* больше или равно значению переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.

Опера-тор	Назначение	Пример	Результат оценки
AND (&)	Логическое И	a AND b a & b	Результатом является логическое И переменных *a* и *b* типа BOOL.
OR	Логическое ИЛИ	a OR b	Результатом является логическое ИЛИ переменных *a* и *b* типа BOOL.
XOR	Исключающее ИЛИ	a XOR b	Результатом является логическое исключающее ИЛИ переменных *a* и *b* типа BOOL.
NOT	НЕ	NOT a	Результатом является отрицание (NOT) переменной *a* типа BOOL.

На показанной ниже блок-схеме конструкция IF выполняется на основе результатов оценки выражений условия 1 и 2. Как показано ниже, в конструкции IF можно использовать более одной инструкции.

```
IF выражение условия 1 THEN
    инструкция 1;
ELSIF выражение условия 2 THEN
    инструкция 2;
ELSE
    инструкция 3;
END_IF;
```



Дополнительная информация

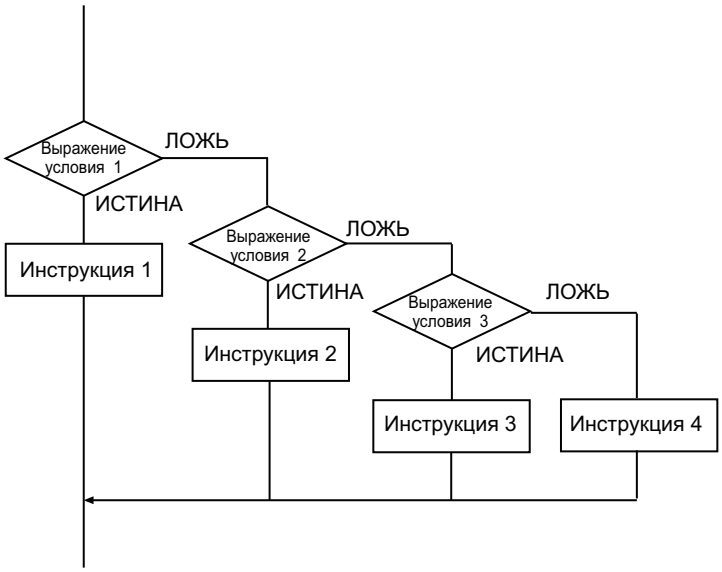
- Инструкции IF можно вкладывать друг в друга. В представленном ниже примере инструкция 11 выполняется, если результатом оценки обоих выражений условия (1 и 11) является ИСТИНА.

```
IF выражение условия 1 THEN
    IF выражение условия 11 THEN
        инструкция 11;
    ELSIF выражение условия 12 THEN
        инструкция 12;
    ELSE
        инструкция 13;
    END_IF;
```

```
ELSIF выражение условия 2 THEN
    инструкция 2;
ELSE
    инструкция 3;
END_IF;
```

Конструкцию ELSIF можно использовать более одного раза. Пример и блок-схема для этого случая приведены ниже.

```
IF выражение условия 1 THEN
    инструкция 1;
ELSIF выражение условия 2 THEN
    инструкция 2;
ELSIF выражение условия 3 THEN
    инструкция 3;
ELSE
    инструкция 4;
END_IF;
```



- В том случае, когда в конструкции IF используется только одно выражение условия, ключевое слово ELSIF не требуется. Также не требуется ключевое слово ELSE, если не требуется выполнять никаких операций, когда ни одно из выражений условия не дает результат ИСТИНА. Пример и блок-схема для этого случая приведены ниже.

```
IF выражение условия THEN
    инструкция;
END_IF;
```

- Каких-либо ограничений в отношении использования инструкций не предусмотрено. В конструкции IF, как и в других командах, можно использовать любые инструкции, например вызовы функциональных блоков и инструкции FOR.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Ключевое слово IF всегда должно использоваться вместе с ключевым словом END_IF. Ключевые слова IF и END_IF должны использоваться в паре.
- Допускается не более 15 уровней вложения конструкций, включая конструкции IF, CASE, FOR, WHILE и REPEAT.

Пример программы

В приведенном ниже примере значение INT#0 присваивается переменной *def*, если значение переменной *abc* меньше INT#0. Если же значение переменной *abc* равно INT#0, то переменной *def* присваивается значение INT#1, а переменной *ghi* присваивается значение INT#2. Наконец, если значение переменной *abc* не равно ни тому, ни другому, то переменной *def* присваивается значение INT#3.

Переменная	Тип данных	Начальное значение
abc	INT	0
def	INT	0
ghi	INT	0

```
IF (abc<INT#0) THEN
    def:=INT#0;
ELSIF (abc=INT#0) THEN
    def:=INT#1;
    ghi:=INT#2;
ELSE
    def:=INT#3;
END_IF;
```

CASE

Конструкция CASE выбирает для выполнения одну из нескольких инструкций в зависимости от значения указанного целочисленного выражения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CASE	Case	---	Нет	CASE целочисленное выражение OF значение: инструкция; значение: инструкция; : ELSE инструкция; END_CASE;

Переменные

Нет

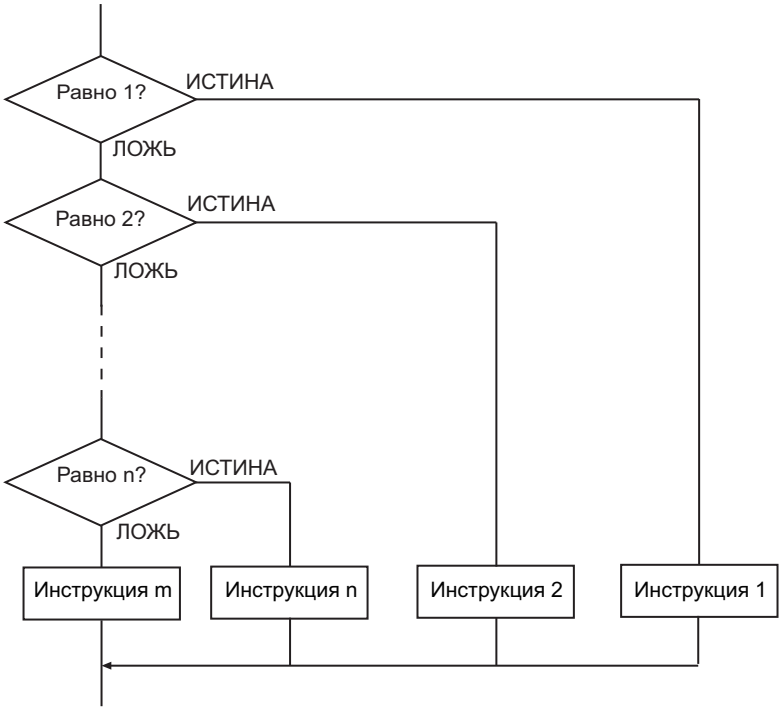
Функция

Конструкция CASE выбирает для выполнения одну из нескольких инструкций в зависимости от значения указанного целочисленного выражения.
Допускается использовать указанные ниже целочисленные выражения и значения.

	Допустимая форма записи
Целочисленное выражение	Целочисленная переменная, целочисленная константа, целочисленное выражение или функция, которые возвращают целочисленное значение, переменная типа «перечисление», выражение с использованием перечисления или перечислитель
Значения	Целочисленные константы

В представленном ниже примере приведена блок-схема алгоритма обработки целочисленного выражения. Как видно из примера, допускается использовать более одной инструкции.

```
CASE целочисленное выражение OF
    1 :
        инструкция 1;
    2 :
        инструкция 2;
    :
    n :
        инструкция n;
    ELSE
        инструкция m;
END_CASE;
```



Дополнительная информация

- Инструкции CASE можно вкладывать друг в друга. В следующем примере выполняется инструкция 12, если значение целочисленного выражения 1 равно 1, а значение целочисленного выражения 11 равно 2.

```
CASE целочисленное выражение 1 OF
    1:
        CASE целочисленное выражение 1 OF
            1:
                инструкция 11;
            2:
                инструкция 12;
            ELSE
                инструкция 1m;
        END_CASE;
    2:
        инструкция 2;
    3:
        инструкция 3;
    ELSE
        инструкция m;
END_CASE;
```

- Можно использовать более одного значения одновременно. Значения следует разделять запятыми. В следующем примере инструкция 1 выполняется, если значение целочисленного выражения равно 1 или 2.

```
CASE целочисленное выражение 1 OF
    1,2:
        инструкция 1;
```

```
    3:
        инструкция 2;
    4:
        инструкция 3;
    ELSE
        инструкция m;
END_CASE;
```

- Можно использовать ряд последовательных значений. Чтобы задать диапазон значений, следует разделить два числа двумя точками. В следующем примере инструкция 1 выполняется, если значение целочисленного выражения находится в диапазоне от 10 до 15 включительно.

```
CASE целочисленное выражение 1 OF
    10..15:
        инструкция 1;
    16:
        инструкция 2;
    17:
        инструкция 3;
    ELSE
        инструкция m;
END_CASE;
```

- Ключевое слово ELSE можно опустить. В этом случае ни одна из инструкций не будет выполнена, если ни одно из значений не будет равно значению целочисленного выражения.
- Каких-либо ограничений в отношении использования инструкций не предусмотрено. В конструкции CASE, как и в других командах, можно использовать любые инструкции, например вызовы функциональных блоков и инструкции FOR.
- Работа инструкции CASE отличается от работы оператора условия *switch* языка программирования Си. В случае оператора *switch* языка Си выполняются все инструкции, расположенные после значения, которое совпадает с целочисленным выражением, если только не применяется инструкция *break*. В случае инструкции CASE выполняются только инструкции, выбранные на основе значения, которое совпадает с целочисленным выражением. Например, в показанной ниже программе на языке Си будут выполнены инструкции 1–3 в составе оператора *switch*. В то время как команда CASE выполнит только инструкцию 1.

Оператор switch языка Си

```
val=1;
switch val
{
        case 1:
    statement 1;
        case 2:
    statement 2;
        case 3:
    statement 3;
}
```

Команда CASE

```
val:=1;
CASE val OF
    1:
        statement 1;
    2:
        statement 2;
    3:
        statement 3;
END_CASE;
```

Меры предосторожности для обеспечения надлежащей эксплуатации

- Ключевое слово CASE всегда должно использоваться вместе с ключевым словом END_CASE. Ключевые слова CASE и END_CASE должны использоваться в паре.
- Тип данных целочисленного выражения может отличаться от типа данных значений.

- Каждое значение может быть указано только один раз.
- Допускается не более 15 уровней вложения конструкций, включая конструкции IF, CASE, FOR, WHILE и REPEAT.

Пример программы

Если переменная *abc* содержит значение INT#1, то переменной *def* присваивается значение INT#10. Аналогичным образом, в случае значения INT#2 присваивается значение INT#20, а в случае значения INT#3 присваивается значение INT#30. При любом другом значении присваивается значение переменной *ghi*.

Переменная	Тип данных	Начальное значение
abc	INT	0
def	INT	0
ghi	INT	0

```
CASE abc OF
    INT#1:
        def:=INT#10;
    INT#2:
        def:=INT#20;
    INT#3:
        def:=INT#30;
    ELSE
        def:=ghi;
END_CASE;
```

Если переменная *abc* содержит значение INT#1, то переменной *def* присваивается значение INT#10. Аналогичным образом, в случае значений INT#2 или INT#5 присваивается значение INT#20, а в случае значения в диапазоне от INT#6 до INT#10 включительно присваивается значение INT#30. При любом другом значении никакое значение не присваивается.

Переменная	Тип данных	Начальное значение
abc	INT	0
def	INT	0

```
CASE abc OF
    INT#1:
        def:=INT#10;
    INT#2,INT#5:
        def:=INT#20;
    INT#6..INT#10:
        def:=INT#30;
END_CASE;
```

WHILE

Конструкция WHILE повторно выполняет некоторую инструкцию до тех пор, пока результат оценки указанного выражения условия равен ИСТИНА.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
WHILE	Цикл While	---	Нет	WHILE выражение условия DO инструкция; END_WHILE;

Переменные

Нет

Функция

Конструкция WHILE повторно выполняет некоторую инструкцию до тех пор, пока результат оценки указанного выражения условия равен ИСТИНА. Используемое выражение условия должно возвращать состояние ИСТИНА или ЛОЖЬ, как показано в таблице ниже.

Элемент, используемый для выражения условия	Пример	Результат оценки
Логическое выражение	a > 3	Если значение переменной *a* больше 3, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
	a = b	Если значения переменных *a* и *b* равны, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
Переменная типа BOOL	abc	Если значение переменной *abc* = ИСТИНА, то результат = ИСТИНА. Если значение = ЛОЖЬ, то результат = ЛОЖЬ.
Константа типа BOOL	ИСТИНА	ИСТИНА
Функция с возвращаемым значением типа BOOL	Имя функции	Если функция возвращает ИСТИНА, то результат = ИСТИНА. Если она возвращает ЛОЖЬ, то результат = ЛОЖЬ.

В логическом выражении можно использовать перечисленные ниже операторы.

Оператор	Назначение	Пример	Результат оценки
=	Равно	a = b	Если значения переменных *a* и *b* равны, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
<>	Не равно	a <> b	Если значения переменных *a* и *b* не равны, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
<	Сравнение	a < b	Если значение переменной *a* меньше значения переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
<=		a <= b	Если значение переменной *a* меньше или равно значению переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
>		a > b	Если значение переменной *a* больше значения переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
>=		a >= b	Если значение переменной *a* больше или равно значению переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
AND (&)	Логическое И	a AND b a & b	Результатом является логическое И переменных *a* и *b* типа BOOL.
OR	Логическое ИЛИ	a OR b	Результатом является логическое ИЛИ переменных *a* и *b* типа BOOL.

Опера-тор	Назначение	Пример	Результат оценки
XOR	Исключающее ИЛИ	a XOR b	Результатом является логическое исключающее ИЛИ переменных *a* и *b* типа BOOL.
NOT	НЕ	NOT a	Результатом является отрицание (NOT) переменной *a* типа BOOL.

Пример и блок-схема для этого случая приведены ниже. Допускается использовать более одной инструкции.

```
WHILE выражение условия DO
    инструкция;
END_WHILE;
```

Дополнительная информация

- Если первое выражение условия равно ЛОЖЬ, следующая за ним инструкция не выполняется.
- Каких-либо ограничений в отношении использования инструкций не предусмотрено. В конструкции WHILE, как и в других командах, можно использовать любые инструкции, например вызовы функциональных блоков и инструкции FOR.
- Для отмены циклической обработки должна быть выполнена команда EXIT. Операции, расположенные между командами EXIT и END_WHILE, при этом выполнены не будут.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Ключевое слово WHILE всегда должно использоваться вместе с ключевым словом END_WHILE. Ключевые слова WHILE и END_WHILE должны использоваться в паре.
- Допускается не более 15 уровней вложения конструкций, включая конструкции IF, CASE, FOR, WHILE и REPEAT.

Пример программы

Значение INT#7 циклически добавляется к переменной *abc* до тех пор, пока значение переменной *abc* меньше или равно INT#1000.

Переменная	Тип данных	Начальное значение
abc	INT	0

```
abc:=INT#0;
WHILE abc<=INT#1000 DO
    abc:=abc+INT#7;
END_WHILE;
```

REPEAT

Конструкция REPEAT выполняет некоторую инструкцию один раз, а затем выполняет ее повторно до тех пор, пока указанное выражение условия не становится равным ИСТИНА.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
REPEAT	Цикл Repeat	---	Нет	REPEAT инструкция; UNTIL выражение условия END_REPEAT;

Переменные

Нет

Функция

Конструкция REPEAT выполняет некоторую инструкцию один раз, а затем выполняет ее повторно до тех пор, пока указанное выражение условия не становится равным ИСТИНА. Используемое выражение условия должно возвращать состояние ИСТИНА или ЛОЖЬ, как показано в таблице ниже.

Элемент, используемый для выражения условия	Пример	Результат оценки
Логическое выражение	a > 3	Если значение переменной *a* больше 3, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
	a = b	Если значения переменных *a* и *b* равны, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
Переменная типа BOOL	abc	Если значение переменной *abc* = ИСТИНА, то результат = ИСТИНА. Если значение = ЛОЖЬ, то результат = ЛОЖЬ.
Константа типа BOOL	ИСТИНА	ИСТИНА
Функция с возвращаемым значением типа BOOL	Имя функции	Если функция возвращает ИСТИНА, то результат = ИСТИНА. Если она возвращает ЛОЖЬ, то результат = ЛОЖЬ.

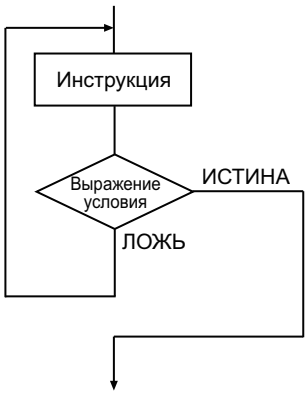
В логическом выражении можно использовать перечисленные ниже операторы.

Оператор	Назначение	Пример	Результат оценки
=	Равно	a = b	Если значения переменных *a* и *b* равны, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
<>	Не равно	a <> b	Если значения переменных *a* и *b* не равны, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
<	Сравнение	a < b	Если значение переменной *a* меньше значения переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
<=		a <= b	Если значение переменной *a* меньше или равно значению переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
>		a > b	Если значение переменной *a* больше значения переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.
>=		a >= b	Если значение переменной *a* больше или равно значению переменной *b*, результат = ИСТИНА. В противном случае результат = ЛОЖЬ.

Оператор	Назначение	Пример	Результат оценки
AND (&)	Логическое И	a AND b a & b	Результатом является логическое И переменных *a* и *b* типа BOOL.
OR	Логическое ИЛИ	a OR b	Результатом является логическое ИЛИ переменных *a* и *b* типа BOOL.
XOR	Исключающее ИЛИ	a XOR b	Результатом является логическое исключающее ИЛИ переменных *a* и *b* типа BOOL.
NOT	НЕ	NOT a	Результатом является отрицание (NOT) переменной *a* типа BOOL.

Пример и блок-схема для этого случая приведены ниже. Допускается использовать более одной инструкции.

```
REPEAT
    инструкция;
UNTIL выражение условия
END_REPEAT;
```



Дополнительная информация

- Перед оценкой выражения условия инструкция выполняется один раз. Следовательно, инструкция всегда выполняется хотя бы один раз.
- Каких-либо ограничений в отношении использования инструкций не предусмотрено. В конструкции REPEAT, как и в других командах, можно использовать любые инструкции, например вызовы функциональных блоков и инструкции FOR.
- Для отмены циклической обработки должна быть выполнена команда EXIT. Операции, расположенные между командами EXIT и END_REPEAT, при этом выполнены не будут.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Ключевое слово REPEAT всегда должно использоваться вместе с ключевыми словами UNTIL и END_REPEAT. Ключевые слова REPEAT, UNTIL и END_REPEAT составляют единую конструкцию.
- Допускается не более 15 уровней вложения конструкций, включая конструкции IF, CASE, FOR, WHILE и REPEAT.

Пример программы

Значение INT#1 циклически добавляется к переменной *abc* до тех пор, пока значение переменной *abc* не становится больше INT#10.

Переменная	Тип данных	Начальное значение
abc	INT	0

```
abc:=INT#0;
REPEAT
    abc:=abc+INT#1;
UNTIL abc>INT#10
END_REPEAT;
```

EXIT

Команда EXIT завершает выполнение самого внутреннего цикла, созданного командой циклической обработки FOR, WHILE или REPEAT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST

EXIT	Выход из цикла	---	Нет	FOR Index:=0 TO 9 BY 1 DO
				  IF Error[Index] THEN
				  EXIT;
				  END_IF;
				END_FOR;

Переменные

Нет

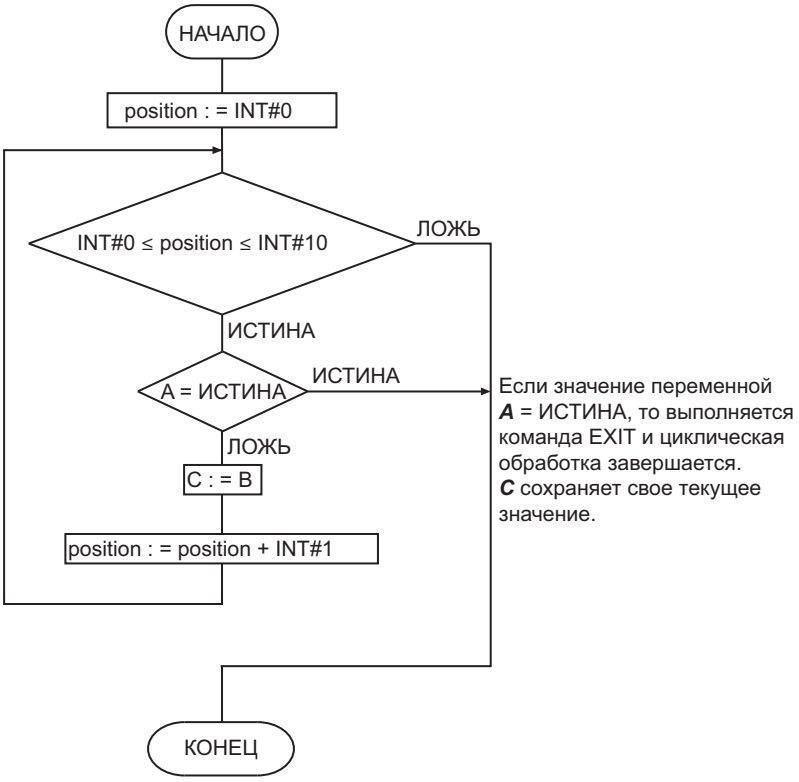
Функция

Команда EXIT завершает выполнение самого внутреннего цикла, созданного командой циклической обработки FOR, WHILE или REPEAT. Управление переходит к команде, которая следует по порядку за прерванным циклом.

В приведенном ниже примере программы в начале выполнения каждого цикла команды FOR проверяется значение переменной *A*. Если значение переменной *A* равно ИСТИНА, то выполняется команда EXIT и циклическая обработка завершается. Если это происходит, то инструкция *C:=B;*, следующая за ключевым словом END_IF, не выполняется, и сохраняется предыдущее значение переменной *C*.

```
FOR position:=INT#0 TO INT#10 BY INT#1 DO
    IF (A=TRUE) THEN
        EXIT;
    END_IF;
    C:=B;
END_FOR;
```

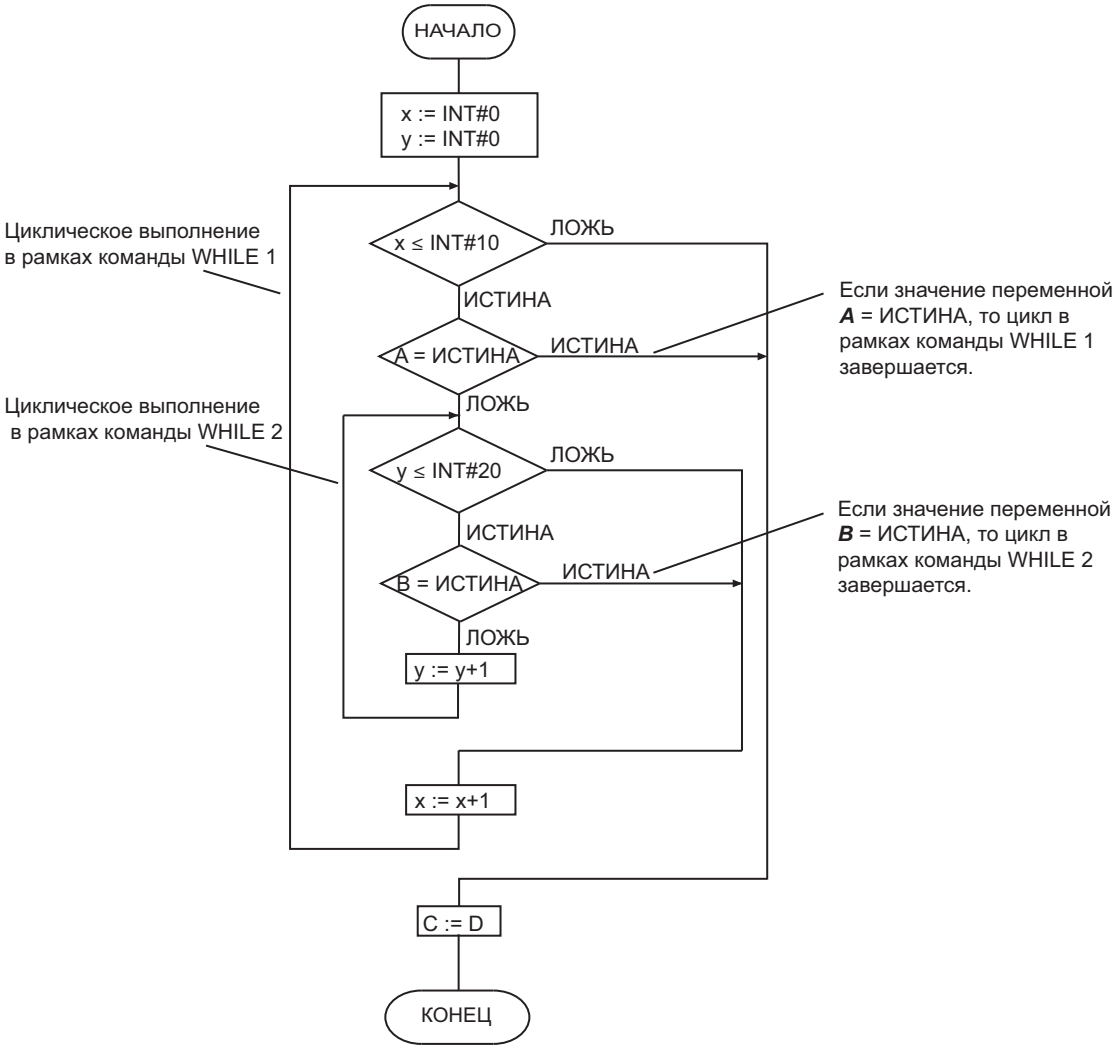
Ниже приведена блок-схема для данной программы.



При выполнении команды EXIT завершается выполнение только самого внутреннего цикла. Если в показанной ниже программе значение переменной *B* равно ИСТИНА, то выполняется 2-я команда EXIT и завершается выполнение цикла, который соответствует 2-й команде WHILE. После этого выполняется инструкция *x:=x+1;*. Выполнение цикла, созданного 1-й командой WHILE (на один уровень выше), в этом случае продолжается.
Если значение переменной *A* равно ИСТИНА, то выполняется 1-я команда EXIT и завершается цикл, созданный 1-й командой WHILE. Далее выполняется инструкция *C:=D;*.

```
x:=INT#0;
y:=INT#0;
WHILE x<=INT#10 DO        // 1-я команда WHILE
    IF (A=TRUE) THEN
        EXIT;             // 1-я команда EXIT
    END_IF;
    WHILE y<=INT#20 DO    // 2-я команда WHILE
        IF (B=TRUE) THEN
            EXIT;         // 2-я команда EXIT
        END_IF;
        y:=y+1;
    END_WHILE;
    x=x+1;
END_WHILE
C:=D;
```

Ниже приведена блок-схема для данной программы.



Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда размещайте эту команду между командами FOR и END_FOR, WHILE и END_WHILE или REPEAT и END_REPEAT.
- Чтобы полностью завершить циклическую обработку в случае вложения циклов, для каждого уровня вложения требуется предусмотреть одну команду EXIT.

RETURN

Описание данной команды см. в описании команды *RETURN* на стр. 2-79 в главе «Команды управления последовательностью выполнения».

FOR

Описание данной команды см. в описании команд *FOR и NEXT* на стр. 2-96 в главе «Команды управления последовательностью выполнения».

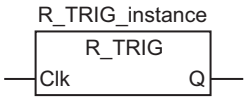
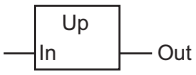
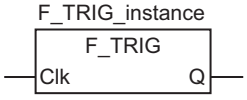
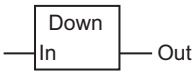
Входные битовые команды

Команда	Имя	Стр.
R_TRIG (Up) и F_TRIG (Down)	Триггер по полож. фронту / Триггер по отриц. фронту	стр. 2-50
TestABit и TestABitN	Проверка бита A / Проверка бита A НЕ	стр. 2-54

R_TRIG (Up) и F_TRIG (Down)

R_TRIG (Up) : Выдает состояние ИСТИНА только в течение одного цикла выполнения задачи, когда состояние входного сигнала меняется на ИСТИНА.

F_TRIG (Down) : Выдает состояние ИСТИНА только в течение одного цикла выполнения задачи, когда состояние входного сигнала меняется на ЛОЖЬ.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
R_TRIG	Триггер по полож. фронту	FB		R_TRIG_instance(Clk, Q);
Up		FUN		Нет
F_TRIG	Триггер по отриц. фронту	FB		F_TRIG_instance(Clk, Q);
Down		FUN		Нет

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Clk, In	Входной сигнал	Вход	Входной сигнал	Зависит от типа данных.	---	---
Q, Out	Выходной сигнал	Выход	Выходной сигнал	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Clk, In	ОК																			
Q, Out	ОК																			

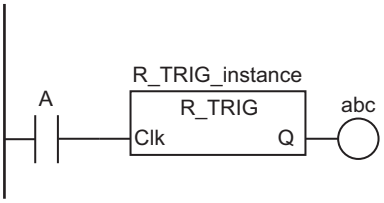
Функция

R_TRIG

Команда R_TRIG переводит выходной сигнал *Q* в состояние ИСТИНА только на один цикл выполнения задачи, когда состояние входного сигнала *Clk* меняется на ИСТИНА. В противном случае *Q* содержит значение ЛОЖЬ.
Команды R_TRIG и Up имеют одинаковое назначение.

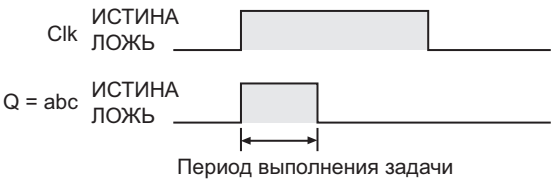
Пример программы и временная диаграмма показаны на рисунке ниже.

LD



ST

```
R_TRIG_instance(A, abc);
```

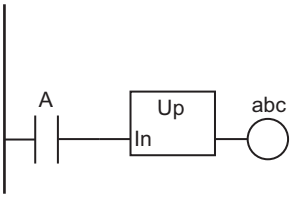


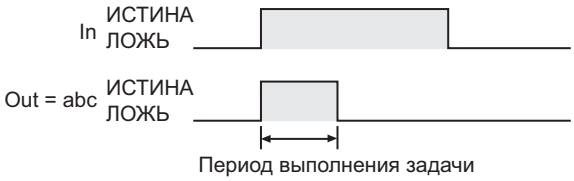
Up

Команда Up переводит выходной сигнал *Out* в состояние ИСТИНА только на один цикл выполнения задачи, когда состояние входного сигнала *In* меняется на ИСТИНА. В противном случае *Out* содержит значение ЛОЖЬ.
Команды R_TRIG и Up имеют одинаковое назначение.
Однако работа команды Up отличается от работы команды R_TRIG в первом цикле выполнения задачи, в котором выполняется команда. О том, как команда Up работает в первом цикле выполнения задачи, в котором она выполняется, см. в параграфе *Меры предосторожности для обеспечения надлежащей эксплуатации* на стр. 2-53.

Пример программы и временная диаграмма показаны на рисунке ниже.

LD

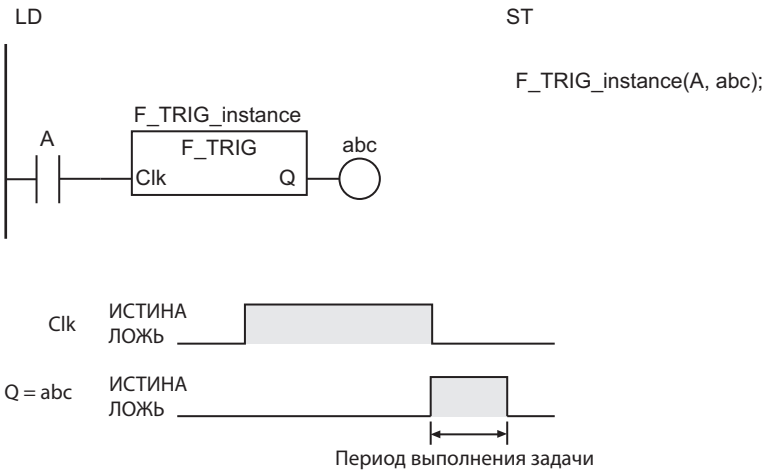




F_TRIG

Команда F_TRIG переводит выходной сигнал *Q* в состояние ИСТИНА только на один цикл выполнения задачи, когда состояние входного сигнала *Clk* меняется на ЛОЖЬ. В противном случае *Q* содержит значение ЛОЖЬ.
Команды F_TRIG и Down работают абсолютно одинаково.

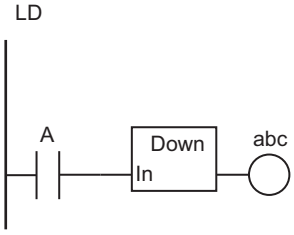
Пример программы и временная диаграмма показаны на рисунке ниже.

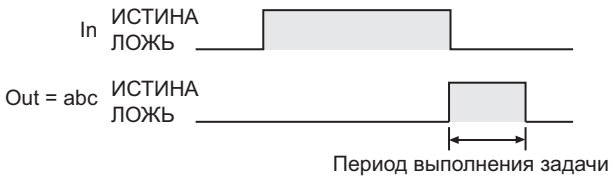


Down

Команда Down переводит выходной сигнал *Out* в состояние ИСТИНА только на один цикл выполнения задачи, когда состояние входного сигнала *In* меняется на ЛОЖЬ. В противном случае *Out* содержит значение ЛОЖЬ.
Команды F_TRIG и Down работают абсолютно одинаково.

Пример программы и временная диаграмма показаны на рисунке ниже.





Меры предосторожности для обеспечения надлежащей эксплуатации

- Обнаружение положительного или отрицательного фронта сигнала зависит от того, как отличается текущее значение сигнала *Clk* или *In* от значения при последнем выполнении команды. Особой осторожности требует ситуация, когда эта команда не выполняется в каждом цикле выполнения задачи из-за применения команды JMP или любой другой команды.
- В случае сбоя питания состояние сигнала *Clk* или *In* не определяется как ЛОЖЬ. Значение сигнала *Clk* или *In* определяется как ЛОЖЬ, только если команда оценивает значение сигнала *Clk* or *In*, когда *Clk* или *In* = ЛОЖЬ.
- В первом цикле выполнения задачи, в котором выполняется команда Up, значение сигнала *Out* всегда равно ЛОЖЬ независимо от значения сигнала *In*.
- Если значение сигнала *In* в команде Up равно ИСТИНА в момент включения источника питания, сигнал *Out* остается в состоянии ЛОЖЬ до тех пор, пока сигнал *In* не принимает значение ЛОЖЬ, а затем снова ИСТИНА.
- В первом цикле выполнения задачи, в котором выполняется команда F_TRIG, значение сигнала *Q* всегда равно ЛОЖЬ независимо от значения сигнала *Clk*.
- Если значение сигнала *Clk* в команде F_TRIG равно ЛОЖЬ в момент включения источника питания, сигнал *Q* остается в состоянии ЛОЖЬ до тех пор, пока сигнал *Clk* не принимает значение ИСТИНА, а затем снова ЛОЖЬ.
- В первом цикле выполнения задачи, в котором выполняется команда Down, значение сигнала *Out* всегда равно ЛОЖЬ независимо от значения сигнала *In*.
- Если значение сигнала *In* в команде Down равно ЛОЖЬ в момент включения источника питания, сигнал *Out* остается в состоянии ЛОЖЬ до тех пор, пока сигнал *In* не принимает значение ИСТИНА, а затем снова ЛОЖЬ.

Информация о версии

Если значение сигнала *Clk* равно ИСТИНА и команда R_TRIG выполняется в один из моментов времени, описанных в таблице ниже, значение сигнала *Q* варьируется в зависимости от версии модуля ЦПУ.

Момент выполнения команды R_TRIG, когда *Clk* = ИСТИНА	Значение *Q*	
	Модуль ЦПУ с версией модуля не ниже 1.02	Модуль ЦПУ с версией модуля не выше 1.01
Период выполнения задачи, в котором R_TRIG выполняется в первый раз	ИСТИНА	Всегда ИСТИНА
При включении источника питания	ИСТИНА	*Q* остается в состоянии ЛОЖЬ до тех пор, пока *Clk* не переходит в состояние ЛОЖЬ, а затем снова в ИСТИНА.

TestABit и TestABitN

TestABit : Выводит значение указанного бита в битовой строке.
TestABitN : Выводит инвертированное значение указанного бита в битовой строке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TestABit	Проверка бита A	FUN		Out:=TestABit (In, Pos);
TestABitN	Проверка бита A НЕ	FUN		Out:=TestABitN (In, Pos);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Битовая строка	Вход	Битовая строка	Зависит от типа данных.	---	*1
Pos	Позиция бита		Указанная позиция бита	От 0 до количества битов в *In* - 1		0
Out	Значение бита	Выход	• TestABit Значение указанного бита • TestABitN Инвертированное значение указанного бита	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

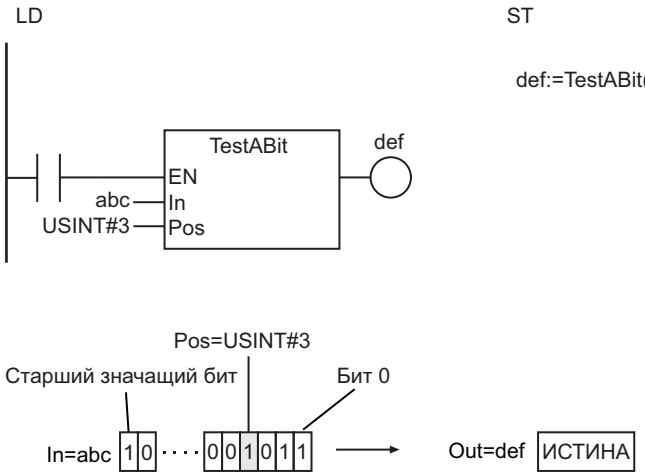
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		ОК	ОК	ОК	ОК															
Pos						ОК														
Out	ОК																			

Функция

TestABit

Команда TestABit считывает состояние бита в позиции *Pos* битовой строки *In* и подает считанное состояние на выход *Out*, когда *EN* равно ИСТИНА.
Когда EN равно ЛОЖЬ, значение *Out* равно ЛОЖЬ.

Ниже показан пример работы команды TestABit для случая, когда *Pos* = *USINT#3*.



TestABitN

Команда TestABitN считывает состояние бита в позиции *Pos* битовой строки *In* и подает инвертированное считанное состояние на выход *Out*, когда *EN* равно ИСТИНА.
Когда EN равно ЛОЖЬ, значение *Out* равно ЛОЖЬ.

Меры предосторожности для обеспечения надлежащей эксплуатации

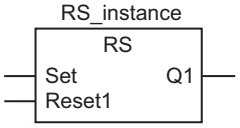
- При использовании команды TestABit или TestABItN в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.
- В указанном ниже случае происходит ошибка. *Out* переходит в состояние ЛОЖЬ.
  a) Значение *Pos* больше, чем количество битов в *In* - 1.

Выходные битовые команды

Команда	Имя	Стр.
RS	Удержание с приоритетом сброса	стр. 2-58
SR	Удержание с приоритетом установки	стр. 2-61
Set и Reset	Установка/Сброс	стр. 2-64
SetBits и ResetBits	Установка битов/Сброс битов	стр. 2-68
SetABit и ResetABit	Установка бита/Сброс бита	стр. 2-71
OutABit	Вывод бита	стр. 2-74

RS

Команда RS хранит значение переменной типа BOOL. Если состояние ИСТИНА одновременно присутствует на входе Set и на входе Reset, приоритетом обладает вход Reset.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
RS	Удержание с приоритетом сброса	FB		RS_instance(Set, Reset1, Q1);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Set *1	Установка	Вход	Вход установки	Зависит от типа данных.	---	0
Reset1 *1	Сброс		Вход сброса			
Q1	Удержание	Выход	Выход удержания	Зависит от типа данных.	---	---

*1. В Sysmac Studio версии 1.03 или более поздней версии есть возможность использовать *S* вместо *Set* и *R1* вместо *Reset1*. Это позволяет нагляднее отображать соответствие между переменными и именами параметров в выражениях языка ST.
Например, можно использовать следующую форму записи: RS_instance(S:=A, R1:=B, Q1=>abc);.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Set	ОК																			
Reset1	ОК																			
Q1	ОК																			

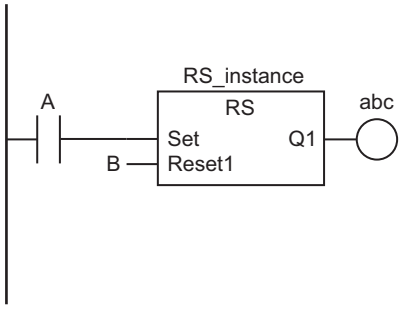
Функция

Команда RS позволяет создать самоблокирующийся выход с приоритетом сигнала сброса. В следующей таблице представлены входные значения и соответствующие им результирующие выходные значения.

Значение *Set*	Значение *Reset1*	Значение *Q1*
ИСТИНА	ИСТИНА	ЛОЖЬ
ИСТИНА	ЛОЖЬ	ИСТИНА
ЛОЖЬ	ИСТИНА	ЛОЖЬ
ЛОЖЬ	ЛОЖЬ	Не изменяется.

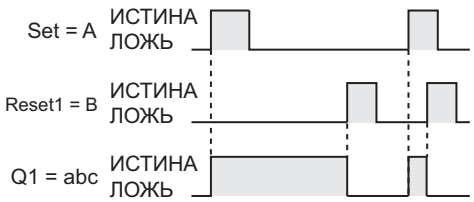
Пример программы и временная диаграмма показаны на рисунке ниже.

LD



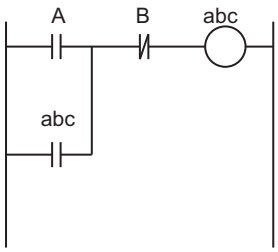
ST

```
RS_instance(A, B, abc);
```



Дополнительная информация

- Команда RS работает аналогично представленной ниже цепи самоблокировки.



- Однако в том случае, когда команда RS находится в области главного управления и область главного управления сброшена, ее работа будет отличаться от работы показанной выше цепи самоблокировки.

Команда/цепь	Значение B	Значение abc
Команда RS	ИСТИНА	Не изменяется.
	ЛОЖЬ	ЛОЖЬ
Цепь самоблокировки	ИСТИНА	ЛОЖЬ
	ЛОЖЬ	

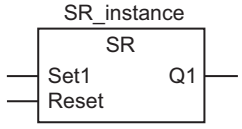
Меры предосторожности для обеспечения надлежащей эксплуатации

- Ни в коем случае не подключайте вход *Reset1* напрямую к нормально замкнутому выходному контакту (биту) внешнего устройства. В случае прерывания (даже кратковременного) питания переменного тока внутренний источник питания в контроллере не выключается сразу и входной сигнал от модуля ввода может сначала перейти во включенное состояние. В результате вход *Reset1* может перейти в состояние ИСТИНА.

- При использовании этой команды в лестничной диаграмме значение выхода *Q1* не изменяется, если в предыдущей команде данной логической цепи происходит ошибка.
- Если эта команда не выполняется из-за выполнения команды перехода (например, команды JMP), на выходе *Q1* сохраняется значение, полученное при последнем выполнении.
- Если эта команда находится в области главного управления и область главного управления сброшена, команда работает следующим образом:
  a) Если значение на входе *Reset1* = ИСТИНА, то на выходе *Q1* сохраняется текущее значение. Если значение Reset1 = ЛОЖЬ, то значение *Q1* меняется на ЛОЖЬ.
  b) Даже если выход *Q1* находится в состоянии ИСТИНА, на вход команды, подключенной к выходу *Q1*, подается ЛОЖЬ.
- Даже если к выходу *Q1* подключен параметр с атрибутом Retain («сохранение»), при отключении питания значение не сохранится. После того как будет возобновлено питание, контроллер перейдет в режим «Выполнение» и будет выполнена данная команда, значение *Q1* поменяется на ЛОЖЬ. В случае применения цепи самоблокировки, приведенной в пункте *Дополнительная информация* на стр. 2-59, значение сохраняется даже после возобновления подачи питания.

SR

Команда SR хранит значение переменной типа BOOL. Если состояние ИСТИНА одновременно присутствует на входе Set и входе Reset, приоритетом обладает вход Set.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SR	Удержание с приоритетом установки	FB		SR_instance(Set1, Reset, Q1);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Set1 *1	Установка	Вход	Вход установки	Зависит от типа данных.	---	0
Reset	Сброс		Вход сброса			
Q1	Удержание	Выход	Выход удержания	Зависит от типа данных.	---	---

*1. В Sysmac Studio версии 1.03 или более поздней версии есть возможность использовать *S1* вместо *Set1* и *R* вместо *Reset*. Это позволяет нагляднее отображать соответствие между переменными и именами параметров в выражениях языка ST.
Например, можно использовать следующую форму записи: SR_instance(S1:=A, R:=B, Q1=>abc);.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Set1	ОК																			
Reset	ОК																			
Q1	ОК																			

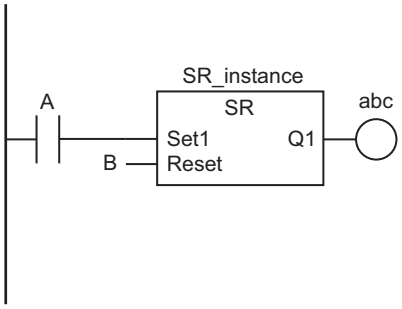
Функция

Команда SR позволяет создать самоблокирующийся выход с приоритетом сигнала установки. В следующей таблице представлены входные значения и соответствующие им результирующие выходные значения.

Значение *Set1*	Значение *Reset*	Значение *Q1*
ИСТИНА	ИСТИНА	ИСТИНА
ИСТИНА	ЛОЖЬ	ИСТИНА
ЛОЖЬ	ИСТИНА	ЛОЖЬ
ЛОЖЬ	ЛОЖЬ	Не изменяется.

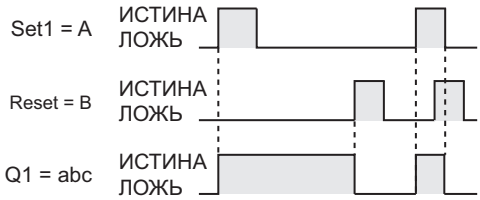
Пример программы и временная диаграмма показаны на рисунке ниже.

LD



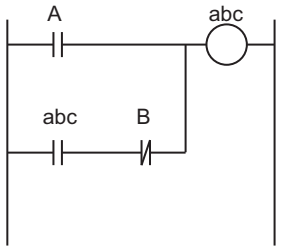
ST

```
SR_instance(A, B, abc);
```



Дополнительная информация

- Команда SR работает аналогично представленной ниже цепи самоблокировки.



- Однако в том случае, когда команда SR находится в области главного управления и область главного управления сброшена, ее работа будет отличаться от работы показанной выше цепи самоблокировки.

Команда/цепь	Значение B	Значение abc
Команда SR	ИСТИНА	Не изменяется.
	ЛОЖЬ	ЛОЖЬ
Цепь самоблокировки	ИСТИНА	ЛОЖЬ
	ЛОЖЬ	

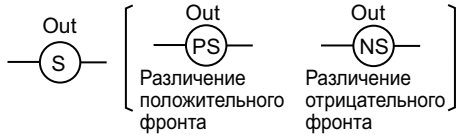
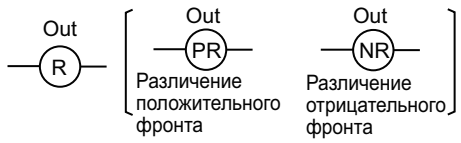
Меры предосторожности для обеспечения надлежащей эксплуатации

- Ни в коем случае не подключайте вход *Reset* напрямую к нормально замкнутому выходному контакту (биту) внешнего устройства. В случае прерывания (даже кратковременного) питания переменного тока внутренний источник питания в контроллере не выключается сразу и входной сигнал от модуля ввода может сначала перейти во включенное состояние. В результате вход *Reset* может перейти в состояние ИСТИНА.
- При использовании этой команды в лестничной диаграмме значение выхода *Q1* не изменяется, если в предыдущей команде данной логической цепи происходит ошибка.

- Если эта команда не выполняется из-за выполнения какой-либо команды перехода (например, команды JMP), на выходе *Q1* сохраняется значение, полученное при последнем выполнении.
- Если эта команда находится в области главного управления и область главного управления сброшена, команда работает следующим образом:
  a) Если значение на входе *Reset* = ИСТИНА, то на выходе *Q1* сохраняется текущее значение. Если на входе ЛОЖЬ, то значение *Q1* меняется на ЛОЖЬ.
  b) Даже если выход *Q1* находится в состоянии ИСТИНА, на вход команды, подключенной к выходу *Q1*, подается ЛОЖЬ.
- Даже если к выходу *Q1* подключен параметр с атрибутом Retain («сохранение»), при отключении питания значение не сохраняется. После того как будет возобновлено питание, контроллер перейдет в режим «Выполнение» и будет выполнена данная команда, значение *Q1* поменяется на ЛОЖЬ. В случае применения цепи самоблокировки, приведенной в пункте «Дополнительная информация», значение сохраняется даже после возобновления подачи питания.

Set и Reset

Set : Переводит переменную типа BOOL в состояние ИСТИНА.
Reset : Переводит переменную типа BOOL в состояние ЛОЖЬ.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Set	Установка	---		Нет
Reset	Сброс	---		Нет

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Выход	Выход	Выход	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	ОК																			

Функция

Set

Команда Set переводит выход *Out* в состояние ИСТИНА, если на входе присутствует состояние ИСТИНА.
После перевода выхода *Out* в состояние ИСТИНА команда Set не переводит выход *Out* в состояние ЛОЖЬ, даже если входное состояние меняется на ЛОЖЬ.
Для сброса выхода *Out* в состояние ЛОЖЬ следует использовать команду Reset.

Ниже показана работа команды для случая, когда не указано выделение положительного или отрицательного фронта.

Команда	Вход	Выход
Set	ИСТИНА	ИСТИНА
	ЛОЖЬ	Не изменяется.

Если указано различение положительного или отрицательного фронта, выполнение команды зависит от следующих факторов: значение на входе при последнем выполнении команды и текущее значение на входе. Сказанное описано в таблице ниже.

Команда	Различение фронта	Значение на входе при последнем выполнении и текущее значение	Выходное значение
Set	Различение положительного фронта	ЛОЖЬ при последнем выполнении -> сейчас ИСТИНА	ИСТИНА
		Прочее, кроме вышеуказанного.	Не изменяется.
	Различение отрицательного фронта	ИСТИНА при последнем выполнении -> сейчас ЛОЖЬ	ИСТИНА
		Прочее, кроме вышеуказанного.	Не изменяется.

Reset

Команда Reset переводит выход *Out* в состояние ЛОЖЬ, если на входе присутствует состояние ИСТИНА.
После перевода выхода *Out* в состояние ЛОЖЬ команда Reset не переводит выход *Out* в состояние ИСТИНА, даже если входное состояние меняется на ЛОЖЬ.
Для перевода выхода *Out* в состояние ИСТИНА следует воспользоваться командой Set.

Ниже показана работа команды для случая, когда не указано выделение положительного или отрицательного фронта.

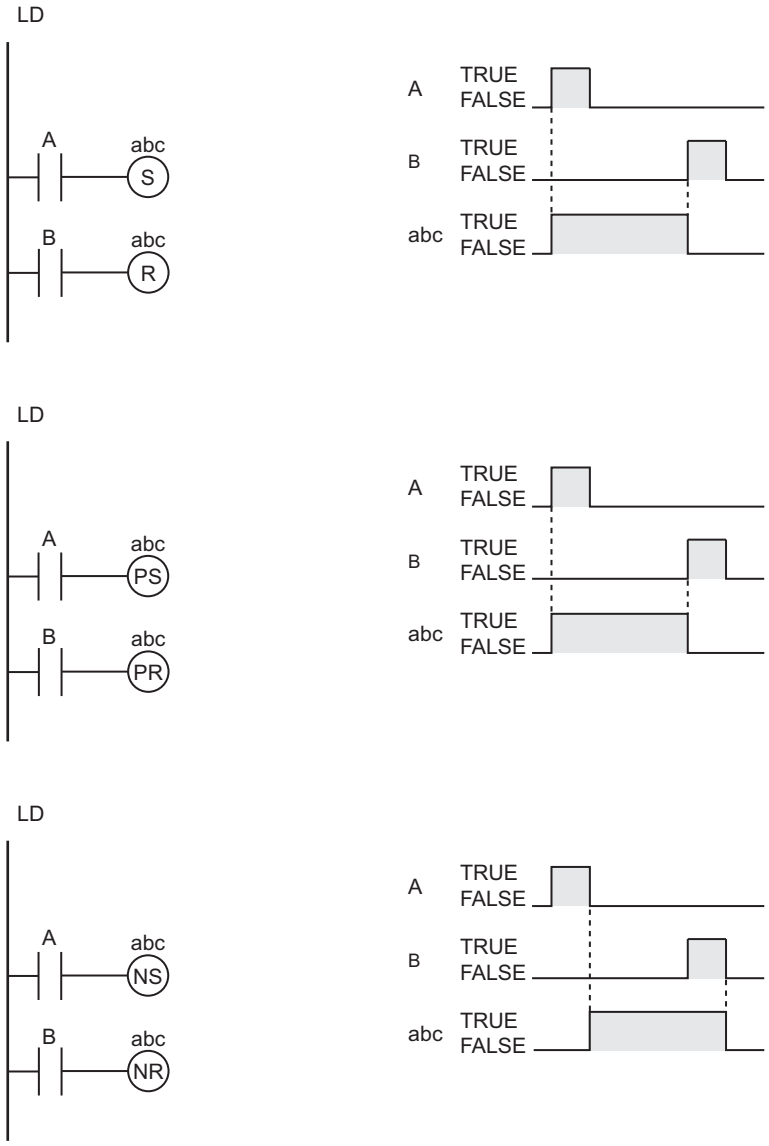
Команда	Вход	Выход
Reset	ИСТИНА	ЛОЖЬ
	ЛОЖЬ	Не изменяется.

Если указано различение положительного или отрицательного фронта, выполнение команды зависит от следующих факторов: значение на входе при последнем выполнении команды и текущее значение на входе. Сказанное описано в таблице ниже.

Команда	Различение фронта	Значение на входе при последнем выполнении и текущее значение	Выходное значение
Reset	Различение положительного фронта	ЛОЖЬ при последнем выполнении -> сейчас ИСТИНА	ЛОЖЬ
		Прочее, кроме вышеуказанного.	Не изменяется.
	Различение отрицательного фронта	ИСТИНА при последнем выполнении -> сейчас ЛОЖЬ	ЛОЖЬ
		Прочее, кроме вышеуказанного.	Не изменяется.

Пример программы и временная диаграмма

Примеры программ и временные диаграммы показаны на рисунках ниже.



Дополнительная информация

Отличия между командами Set и Reset и командой Out

- Команды Set и Reset срабатывают только тогда, когда входное значение меняется на ИСТИНА. Они не срабатывают при входном значении ЛОЖЬ. Другими словами, при входном значении ЛОЖЬ состояние выхода команды не изменяется.
- Команда Out переводит указанную переменную в состояние ИСТИНА, если результатом выполнения предшествующей команды является ИСТИНА, или в состояние ЛОЖЬ, если результатом выполнения предшествующей команды является ЛОЖЬ. То есть она действует как при входном значении ИСТИНА, так и при входном значении ЛОЖЬ.

Отличия между командами Set и Reset и командами SR и RS

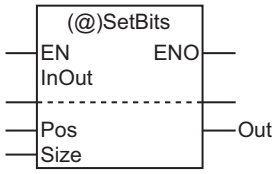
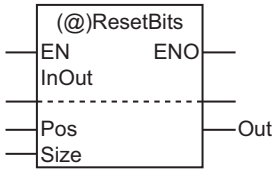
Для команд SR и RS требуется, чтобы входы *Set* и *Reset* находились в одном месте внутри программы. В отличие от них команды Set и Reset можно размещать в разных местах программы.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если команда Set и команда Reset находятся в области главного управления и область главного управления сброшена, значение выхода *Out* не изменяется.
- Если эти команды не выполняются из-за выполнения какой-либо команды перехода (например, команды JMP), значение выхода *Out* не изменяется.
- Если указано выполнение по положительному фронту и в момент включения питания на входе уже присутствует состояние ИСТИНА, эти команды не выполняются. Они будут выполнены, только если их вход перейдет в состояние ЛОЖЬ, а затем вернется в состояние ИСТИНА.
- Если выполнение по положительному фронту не указано и в момент включения питания на входе уже присутствует состояние ИСТИНА, эти команды выполняются. В этом случае нет необходимости переводить вход в состояние ЛОЖЬ перед выполнением команды.

SetBits и ResetBits

SetBits : Переводит группу смежных битов в битовой строке в состояние ИСТИНА.
ResetBits : Переводит группу смежных битов в битовой строке в состояние ЛОЖЬ.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SetBits	Установка битов	FUN		SetBits(InOut, Pos, Size);
ResetBits	Сброс битов	FUN		ResetBits(InOut, Pos, Size);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
InOut	Битовая строка	Вход-выход	Битовая строка	Зависит от типа данных.	---	---
Pos	Позиция бита	Вход	Указанная позиция бита	От 0 до количества битов в *InOut* - 1	---	0
Size	Количество битов		Количество битов	От 0 до количества битов в *InOut*		1
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

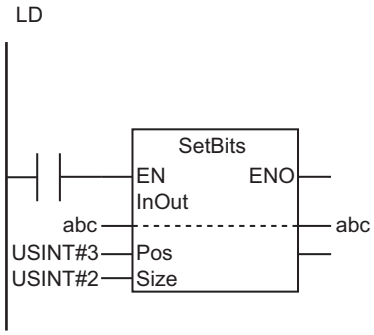
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut		ОК	ОК	ОК	ОК															
Pos						ОК														
Size						ОК														
Out	ОК																			

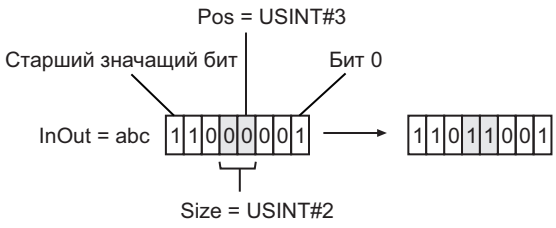
Функция

SetBits

Команда SetBits переводит ряд последовательно расположенных битов в битовой строке *InOut* в состояние ИСТИНА. Количество битов указывается в параметре *Size*, позиция первого бита группы задается параметром *Pos*. Состояния остальных битов при этом не изменяются.

Ниже приведен пример работы команды SetBits для случая, когда *Pos* = *USINT#3*, а *Size* = *USINT#2*.





ResetBits

Команда ResetBits переводит ряд последовательно расположенных битов в битовой строке *InOut* в состояние ЛОЖЬ. Количество битов указывается в параметре *Size*, позиция первого бита группы задается параметром *Pos*. Состояния остальных битов при этом не изменяются.

Дополнительная информация

Эти команды используются, когда переменные с параметром AT, содержащиеся в областях памяти, в которых данные обрабатываются пословно (например, в области DM), нужно глобально перевести в состояние ИСТИНА или ЛОЖЬ.

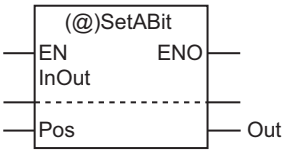
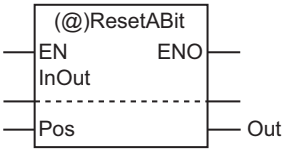
Меры предосторожности для обеспечения надлежащей эксплуатации

- Если команда SetBits и команда ResetBits находятся в области главного управления и область главного управления сброшена, значение выхода *InOut* не изменяется.
- Если эти команды не выполняются из-за выполнения какой-либо команды перехода (например, команды JMP), значение на выходе *InOut* остается прежним.
- Значение выхода *InOut* не изменяется, если *Size* = *0*.

- При использовании этих команд в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значения на выходах *Out* и *InOut* не изменятся.
  a) Значение *Pos* больше, чем количество битов в *InOut* - 1.
  b) Значение *Size* находится за пределами допустимого диапазона.
  c) Значение *Pos* или *Size* больше, чем количество битов в *InOut*.

SetABit и ResetABit

SetABit : Переводит указанный бит в битовой строке в состояние ИСТИНА.
ResetABit : Переводит указанный бит в битовой строке в состояние ЛОЖЬ.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SetABit	Установка бита	FUN		SetABit(InOut, Pos);
ResetABit	Сброс бита	FUN		ResetABit(InOut, Pos);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
InOut	Битовая строка	Вход-выход	Битовая строка	Зависит от типа данных.	---	---
Pos	Позиция бита	Вход	Указанная позиция бита	От 0 до количества битов в *InOut* - 1	---	0
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut		ОК	ОК	ОК	ОК															
Pos						ОК														
Out	ОК																			

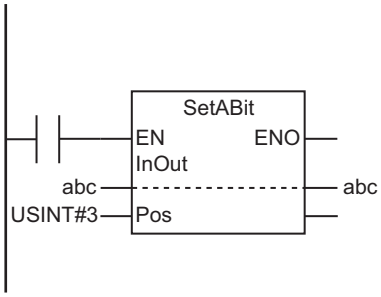
Функция

SetABit

Команда SetABit переводит бит в позиции *Pos* битовой строки *InOut* в состояние ИСТИНА. Биты, которые не указаны, не изменяются.
Даже если после выполнения команды состояние входа *EN* меняется на ЛОЖЬ, состояние бита в позиции *Pos* битовой строки *InOut* не меняется.

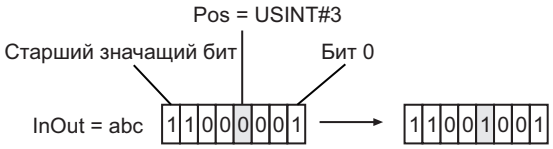
Ниже показан пример работы команды SetABit для случая, когда *Pos* = *USINT#3*.

LD



ST

```
SetABit(abc, USINT#3);
```



ResetABit

Команда ResetABit переводит бит в позиции *Pos* битовой строки *InOut* в состояние ЛОЖЬ. Биты, которые не указаны, не изменяются.
Даже если после выполнения команды состояние входа *EN* меняется на ЛОЖЬ, состояние бита в позиции *Pos* битовой строки *InOut* не меняется.

Дополнительная информация

Отличия между командами SetABit и ResetABit и командой OutABit

- Команды SetABit и ResetABit меняют значение указанного бита на ИСТИНА или ЛОЖЬ.
- В отличие от них, с помощью команды OutABit можно динамически изменять состояние, в которое требуется перевести указанный бит.

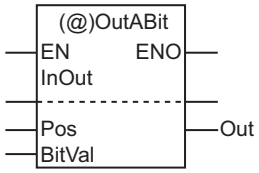
Меры предосторожности для обеспечения надлежащей эксплуатации

- Если команда SetABit и команда ResetABit находятся в области главного управления и область главного управления сброшена, значение выхода *InOut* не изменяется.

- Если эти команды не выполняются из-за выполнения какой-либо команды перехода (например, команды JMP), значение выхода *InOut* не изменяется.
- При использовании этих команд в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значения на выходах *Out* и *InOut* не изменятся.
  a) Значение *Pos* больше, чем количество битов в *In* - 1.

OutABit

Команда OutABit переводит указанный бит в битовой строке в состояние ИСТИНА или ЛОЖЬ.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
OutABit	Вывод бита	FUN		OutABit(InOut, Pos, BitVal);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
InOut	Битовая строка	Вход-выход	Битовая строка	Зависит от типа данных.	---	---
Pos	Позиция бита	Вход	Указанная позиция бита	От 0 до количества битов в *InOut* - 1	---	0
BitVal	Заданное значение		Устанавливаемое значение	Зависит от типа данных.		ИСТИНА
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

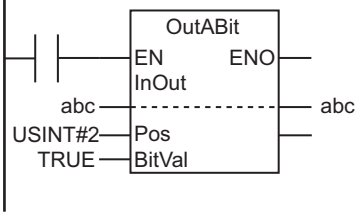
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut		ОК	ОК	ОК	ОК															
Pos						ОК														
BitVal	ОК																			
Out	ОК																			

Функция

Команда OutABit переводит бит в позиции *Pos* битовой строки *InOut* в состояние, заданное параметром *BitVal*. Изменяется только бит в позиции *Pos*.

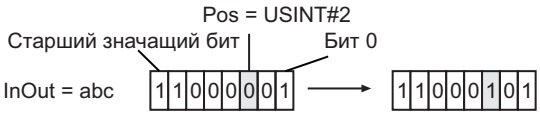
Ниже показан пример для случая, когда *Pos* = *USINT#2*, а *BitVal* = ИСТИНА.

LD



ST

```
OutABit(abc, USINT#2, TRUE);
```



Дополнительная информация

Отличия между командами SetABit и ResetABit и командой OutABit

- Команды SetABit и ResetABit меняют значение указанного бита на ИСТИНА или ЛОЖЬ.
- Что касается команды OutABit, с ее помощью можно динамически изменять значение заданного бита, изменяя значение параметра *BitVal*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если эта команда находится в области главного управления и область главного управления сброшена, значение выхода *InOut* не изменяется.
- Если эта команда не выполняется из-за выполнения какой-либо команды перехода (например, команды JMP), значение выхода *InOut* не изменяется.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значения на выходах *Out* и *InOut* не изменятся.
  a) Значение *Pos* больше, чем количество битов в *InOut* - 1.

Команды управления последовательностью выполнения

Команда	Имя	Стр.
End	Конец	стр. 2-78
RETURN	Возврат	стр. 2-79
MC и MCR	Начало главного управления/Конец главного управления	стр. 2-80
JMP	Переход	стр. 2-94
FOR и NEXT	Начало цикла/Конец цикла	стр. 2-96
BREAK	Выход из цикла	стр. 2-104

End

Команда End завершает выполнение программы в текущем цикле выполнения задачи.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
End	Конец	FUN	End EN ENO	Нет

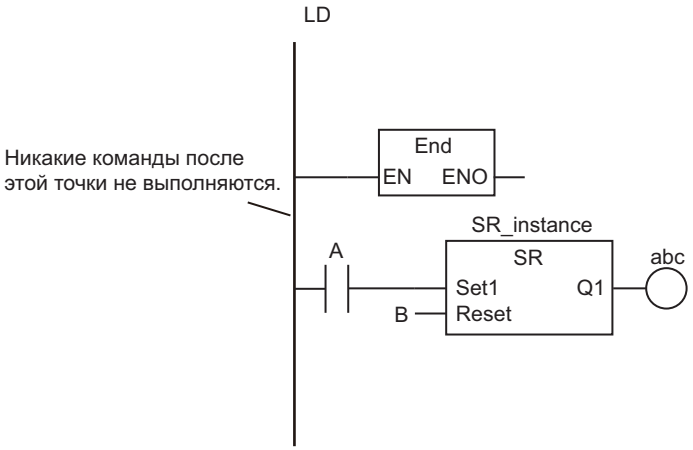
Переменные

Нет

Функция

Команда End завершает выполнение программы в текущем цикле выполнения задачи.

Пример программы представлен на рисунке ниже. После выполнения команды End расположенная после нее команда SR не выполняется.



Меры предосторожности для обеспечения надлежащей эксплуатации

- Эта команда должна использоваться только в программе.
- Если она будет применена в функции, функциональном блоке или вставке на языке ST, произойдет ошибка сборки.
- Эта команда должна подключаться непосредственно к левой шине.

RETURN

Команда RETURN завершает функцию или функциональный блок и возвращает программу к вызвавшей их команде.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
RETURN	Возврат	FUN	RETURN EN ENO	RETURN;

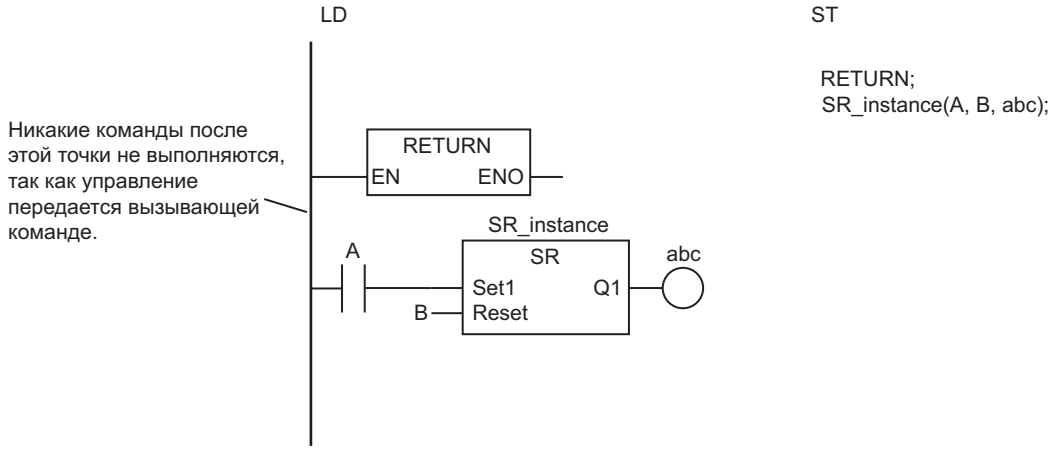
Переменные

Нет

Функция

Команда RETURN завершает функцию или функциональный блок и возвращает программу к вызвавшей их команде.

Пример программы представлен на рисунке ниже. После выполнения команды RETURN расположенная после нее команда SR не выполняется.



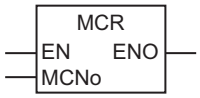
Меры предосторожности для обеспечения надлежащей эксплуатации

- При использовании этой команды в лестничной диаграмме соблюдайте указанные ниже меры предосторожности.
  a) Используйте эту команду только в функциях и функциональных блоках. Если она будет применена в программе, произойдет ошибка сборки.
  b) Всегда подключайте эту команду непосредственно к левой шине лестничной диаграммы.
- Перед выполнением этой команды задайте возвращаемое значение, выходные переменные и значение *ENO* программного компонента.
- Если эта команда используется в программе слишком часто, порядок выполнения программы становится менее понятен. Используйте эту команду с осторожностью.

MC и MCR

MC : Обозначает точку начала области главного управления и сбрасывает область главного управления.

MCR : Обозначает точку конца области главного управления.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MC	Начало главного управления	---		Нет
MCR	Конец главного управления	---		Нет

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In (только команда MC)	Вход главного управления	Вход	ЛОЖЬ: сброс области главного управления.	Зависит от типа данных.	---	---
MCNo	Номер главного управления		Номер главного управления	0...14*1		0

*1. Номер регистрируется в Sysmac Studio автоматически. Задавать его не требуется.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In (только команда MC)	ОК																			
MCNo							ОК													

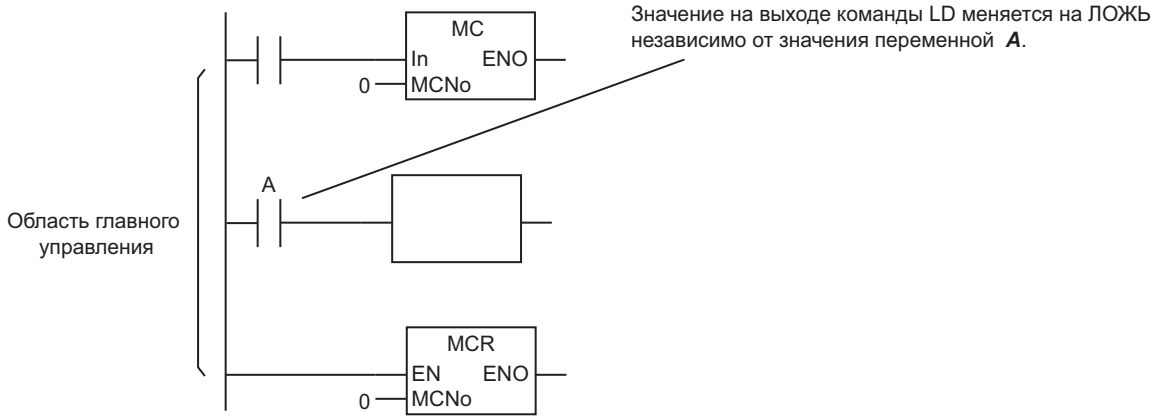
Функция

Функцию главного управления используют для того, чтобы прекратить выполнение или перевести в иное эквивалентное состояние все программные компоненты, расположенные в пределах указанной области программы.

С помощью функции главного управления можно легко управлять условиями выполнения сравнительно длинного сегмента программы.

Область программы, к которой применяется функция главного управления, называют областью главного управления. В начале области главного управления размещают команду MC, а в конце — команду MCR.
Когда состояние входа главного управления *In* меняется на ЛОЖЬ, выходы всех команд LD, подключенных к левой шине в пределах области главного управления, принудительно переводятся в состояние ЛОЖЬ. Это называется сбросом главного управления.

Когда главное управление сброшено, программные компоненты, которые следуют за командами LD, как правило, выполняются так, как если бы условие их выполнения было равно ЛОЖЬ. Некоторые программные компоненты, впрочем, работают по-другому. Это будет пояснено позже.



Если значение на входе *In* равно ИСТИНА, сброс главного управления не выполняется. Программные компоненты в области главного управления работают обычным образом.

Работа программных компонентов во время сброса главного управления

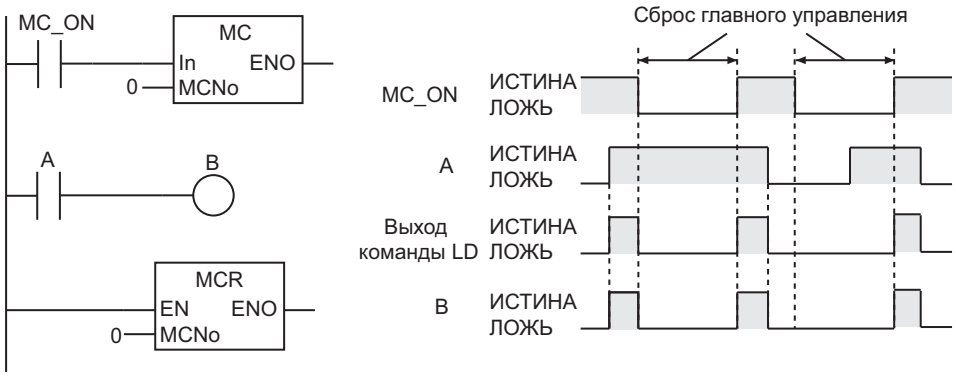
Разные программные компоненты по-разному работают во время сброса главного управления, что поясняется в таблице ниже.

Программный компонент	Работа
Команды Out и OutABit	В указанную переменную выводится ЛОЖЬ.
Команда OutNot	В указанную переменную выводится ЛОЖЬ.
Команды Set и Reset	Сохраняется выходное состояние, действовавшее до сброса главного управления.
Команда TON	Команда выполняется с состоянием ЛОЖЬ на входе таймера *In*. Это означает, что таймер сбрасывается. Значение истекшего времени *ET* становится равно 0, а выход таймера *Q* принимает значение ЛОЖЬ.
Команда TOF	Команда выполняется с состоянием ИСТИНА на входе таймера *In*. Это означает, что таймер сбрасывается. Значение истекшего времени *ET* становится равно 0, а выход таймера *Q* принимает значение ИСТИНА. Однако если к выходу *Q* подключена команда Out, условие выполнения команды Out остается равным ЛОЖЬ.
Команда TP	Команда выполняется с состоянием ЛОЖЬ на входе таймера *In*. Это означает, что таймер сбрасывается.

Программный компонент	Работа
	Таймер активен : Время отсчитывается до конца, после чего значение истекшего времени *ET* обнуляется. До конца отсчета времени значение на выходе таймера *Q* равно ИСТИНА, а затем оно становится равно ЛОЖЬ. Таймер не активен : Значение *ET* становится равно 0, а значение *Q* меняется на ЛОЖЬ. Однако если к выходу *Q* подключена команда Out, условие выполнения команды Out остается равным ЛОЖЬ, даже пока идет отсчет времени (т. е. пока таймер активен).
Команда AccumulationTimer	Команда выполняется с состоянием ЛОЖЬ на входе таймера *In*. Это означает, что таймер прекращает работу. При этом истекшее время *ET* и выход таймера *Q* сохраняют свои текущие значения. Однако если к выходу *Q* подключена команда Out, условие выполнения команды Out остается равным ЛОЖЬ, даже если на выходе *Q* присутствует состояние ИСТИНА. Однако сброс (*Reset*) активируется.
Команда Timer	Команда выполняется с состоянием ЛОЖЬ на входе таймера *In*. Это означает, что таймер сбрасывается. Оставшееся время (*ET*) становится равно уставке таймера (*PT*), а выходное значение таймера (*Q*) меняется на ЛОЖЬ.
Команды CTU, CTD и CTUD	Эти команды не выполняются. Если какая-либо из этих команд выполнялась до сброса главного управления, в ней удерживается результат счета, достигнутый на момент сброса. Если к флагу завершения счетчика *Q* подключена команда Out, условие выполнения команды Out остается равным ЛОЖЬ.
Команда JMP	Эта команда не выполняется.
Команды FOR и NEXT	Эти команды не выполняются.
Команда BREAK	Эта команда не выполняется.
Функциональные блоки, выполняемые в течение нескольких циклов выполнения задачи (т.е. команды с выходными переменными *Done*, *Busy* и *Error*)	Левая шина переходит в состояние ЛОЖЬ (не передает «ток»). Если во время выполнения этой команды будет предпринята попытка сбросить главное управление, команда продолжит выполняться и будет выполнена до конца. На выходы *Busy*, *Done* и *Error* будут подаваться соответствующие сигналы, но если следующей командой является команда вывода, то всегда будет выдаваться значение ЛОЖЬ. Если же к выходу *Busy*, *Done* или *Error* напрямую подключена переменная, в эту переменную будет записываться правильное значение, соответствующее алгоритму работы команды. Значение переменной *Busy*, *Done* или *Error* также можно получить, используя форму записи *имя_экземпляра.выходная_переменная*.
Прочие функции	Эти функции не выполняются.
Прочие функциональные блоки	Левая шина переходит в состояние ЛОЖЬ (не передает «ток»).

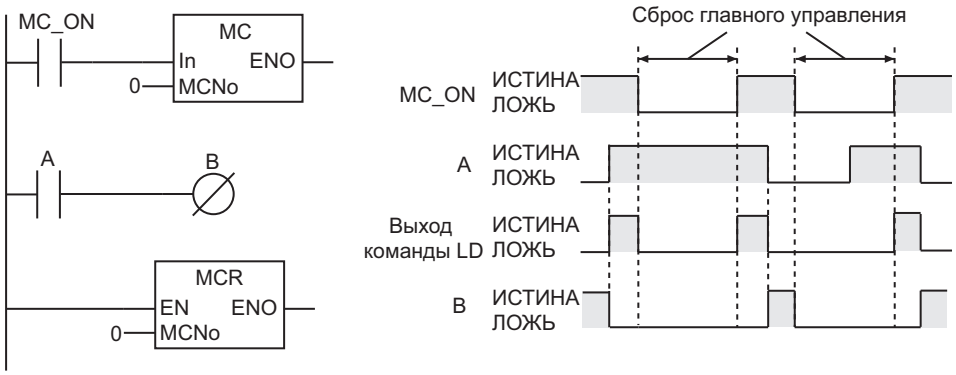
● Out

Во время сброса главного управления выдается значение ЛОЖЬ.



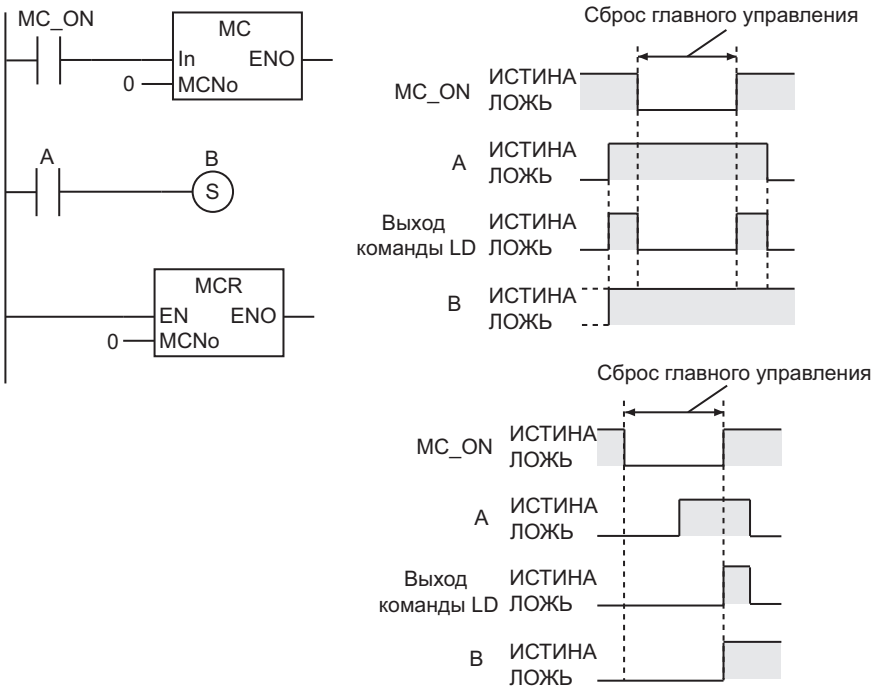
● OutNot

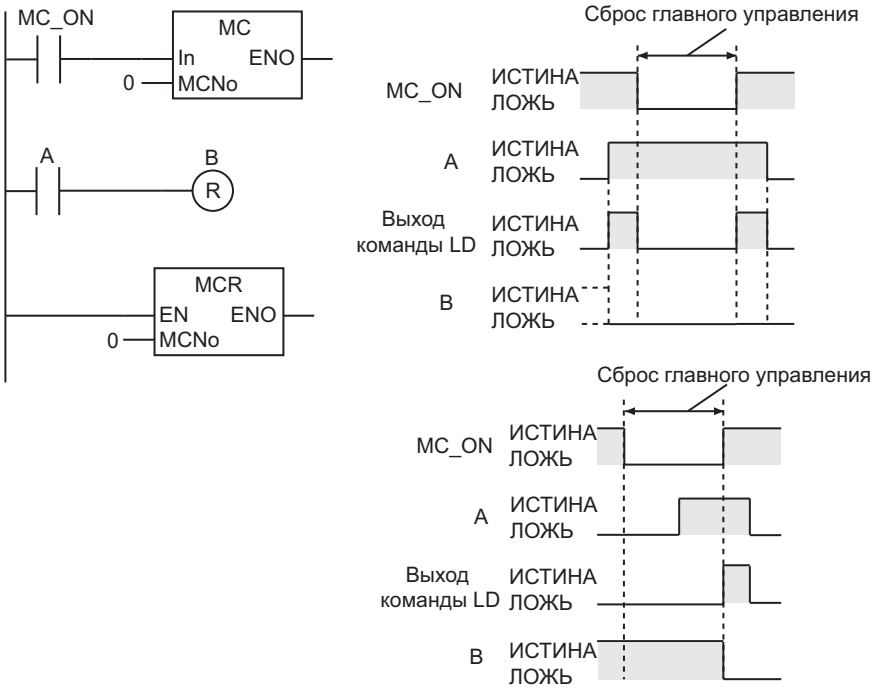
Во время сброса главного управления выдается значение ЛОЖЬ.
Необходимо соблюдать осторожность, так как команда OutNot работает по-другому и сказанное выше не выполняется, когда на выходе предшествующей команды LD присутствует состояние ЛОЖЬ.



● Set и Reset

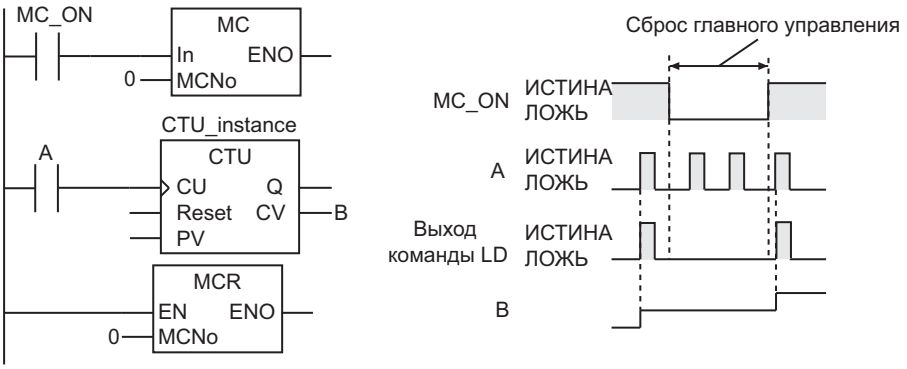
Во время сброса главного управления на выходе сохраняется прежнее значение.





● CTU, CTD и CTUD

Во время сброса главного управления сохраняется предыдущее значение счетчика. После отмены сброса главного управления счет возобновляется с сохраненного значения.



Работа программных компонентов с различением положительного или отрицательного перепада на входе

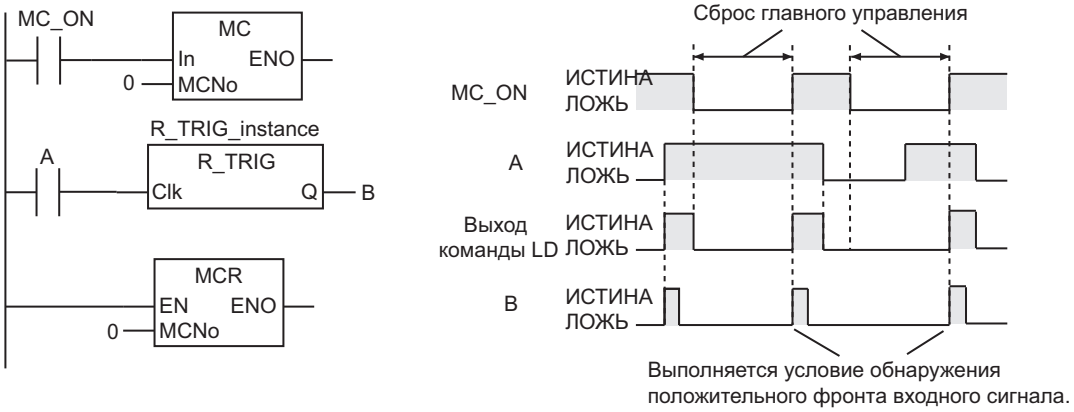
В следующей таблице приведены программные компоненты, для которых может указываться различение положительного или отрицательного фронта входного сигнала.

Различение фронта	Команды
Выделение положительного фронта на входе	• Команды LD, LDN, AND, ANDN, OR, ORN и Out, для которых можно указать выполнение по положительному фронту • R_TRIG (Up) • Функции, для которых можно указать выполнение по положительному фронту с помощью параметра @ • Функциональные блоки (например, команды счетчика), для которых можно указать выполнение по положительному фронту
Выделение отрицательного фронта на входе	• Команды LD, LDN, AND, ANDN, OR, ORN и Out, для которых можно указать выполнение по отрицательному фронту • F_TRIG (Down)

При сбросе главного управления или при отмене сброса условия выполнения этих программных компонентов изменяются. Это означает, что для этих программных компонентов могут оказаться выполнены условия обнаружения положительного или отрицательного перепада. В случае обнаружения положительного или отрицательного перепада эти команды выполняются соответствующим образом.

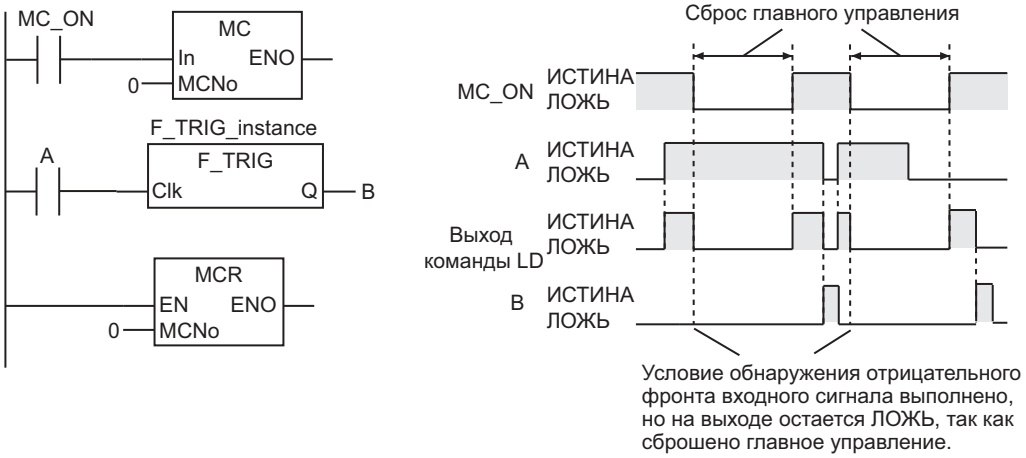
● R_TRIG (Up)

При сбросе главного управления условие выполнения переходит в состояние ЛОЖЬ. Если после отмены сброса главного управления условие выполнения переходит в состоянии ИСТИНА, соблюдается условие положительного перепада на входе и команда выполняется соответствующим образом.



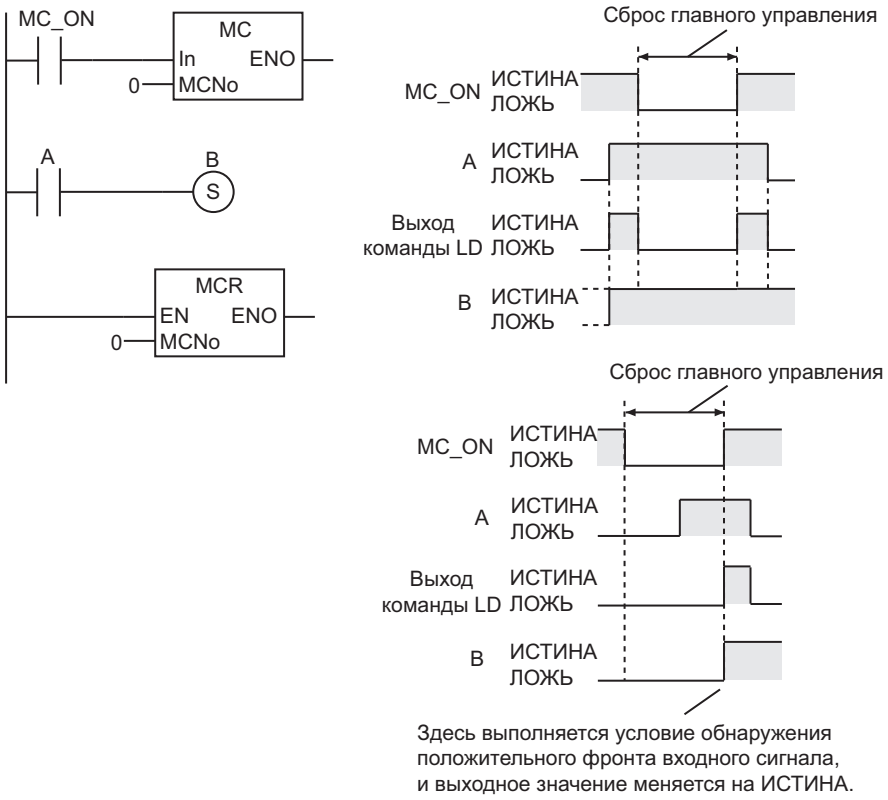
● F_TRIG (Down)

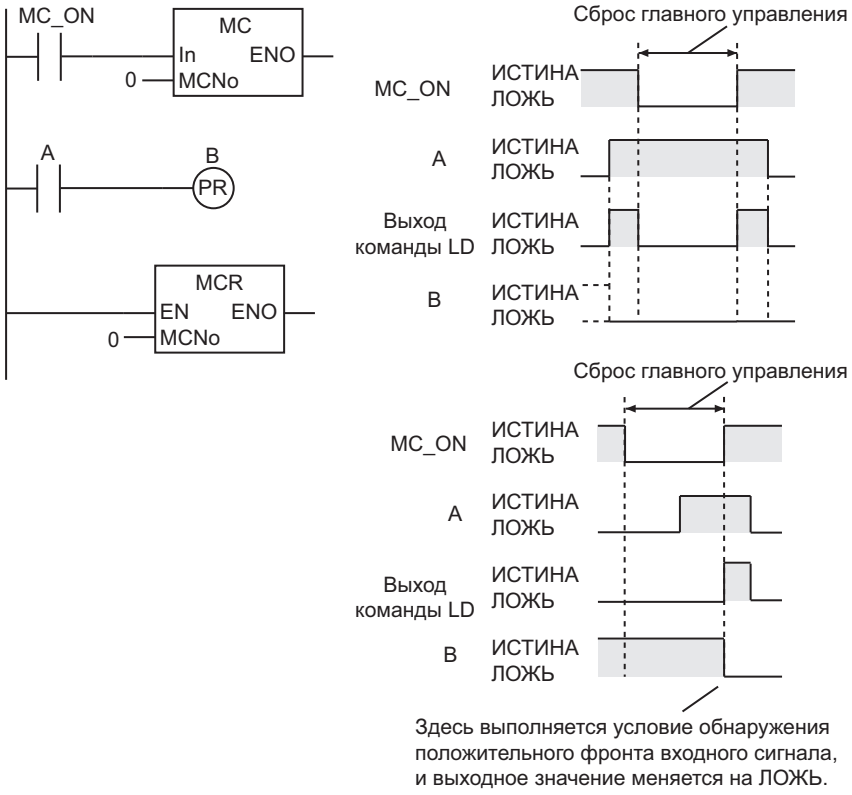
При сбросе главного управления условие выполнения переходит в состояние ЛОЖЬ. Если до этого условие выполнения находилось в состоянии ИСТИНА, оказывается соблюдено условие отрицательного перепада на входе. Однако выход команды F_TRIG (Down) во время сброса главного управления принудительно сбрасывается в состояние ЛОЖЬ, поэтому выходное значение меняется на ЛОЖЬ.



● Set и Reset с различением положительного фронта на входе

Во время сброса главного управления на выходе сохраняется прежнее значение.
После отмены сброса главного управления условие выполнения переходит в состоянии ИСТИНА и команда выполняется.

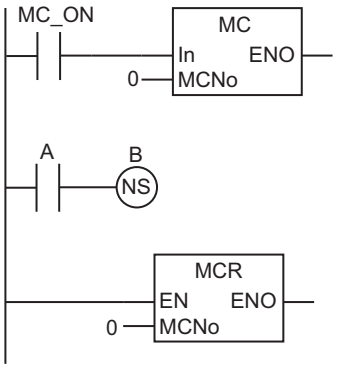


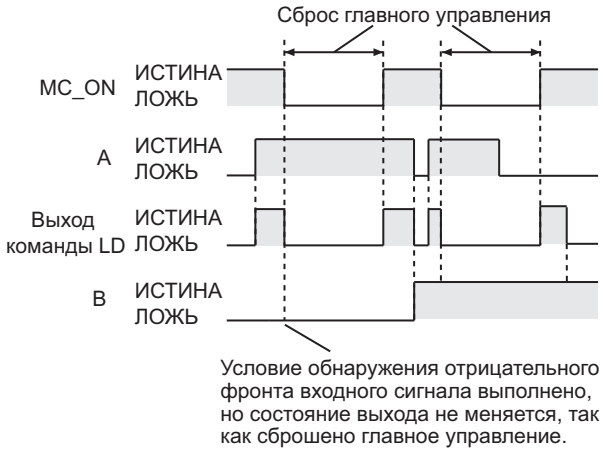


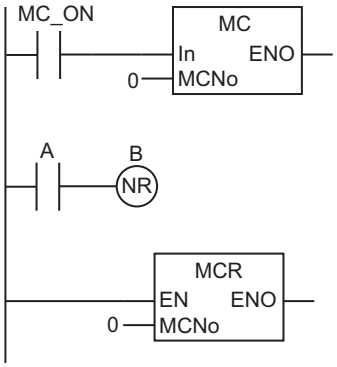
● Set и Reset с различением отрицательного фронта на входе

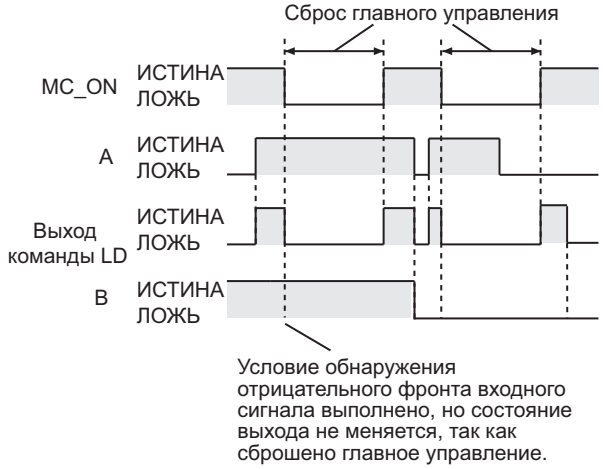
При сбросе главного управления условие выполнения переходит в состояние ЛОЖЬ. Если до этого условие выполнения находилось в состоянии ИСТИНА, оказывается соблюдено условие

отрицательного перепада на входе. Однако во время сброса главного управления на выходе сохраняется предыдущее значение, так что в итоге выходное значение не изменяется.



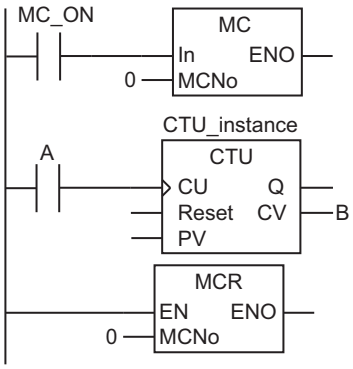


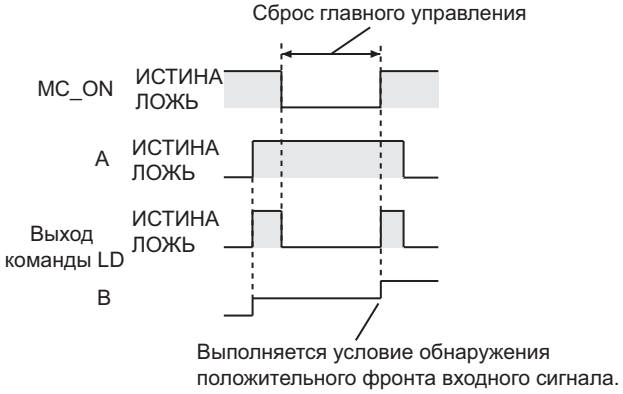




● CTU, CTD и CTUD

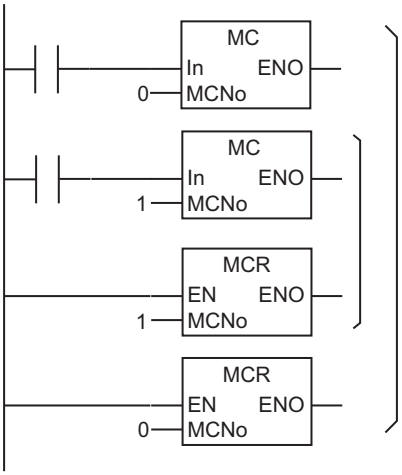
При сбросе главного управления состояние входа счетчика меняется на ЛОЖЬ. Если после отмены сброса главного управления на входе счетчика установится состояние ИСТИНА, то будет соблюдено условие положительного перепада на входе и команда начнет счет.



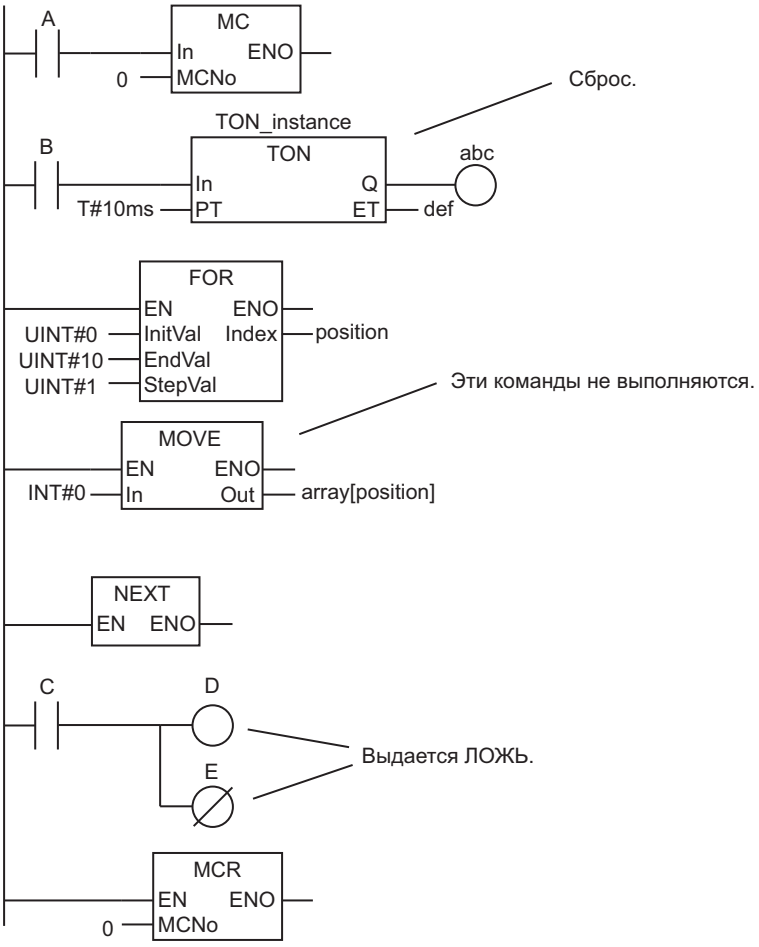


Всегда используйте команды MC и MCR в паре в пределах одного программного компонента. Для команд MC и MCR, работающих в паре, используется одинаковое значение номера главного управления *MCNo*. Значение *MCNo* не может быть задано пользователем. Среда Sysmac Studio регистрирует его автоматически.

Пары команд MC и MCR можно вкладывать друг в друга на глубину до 15 уровней.

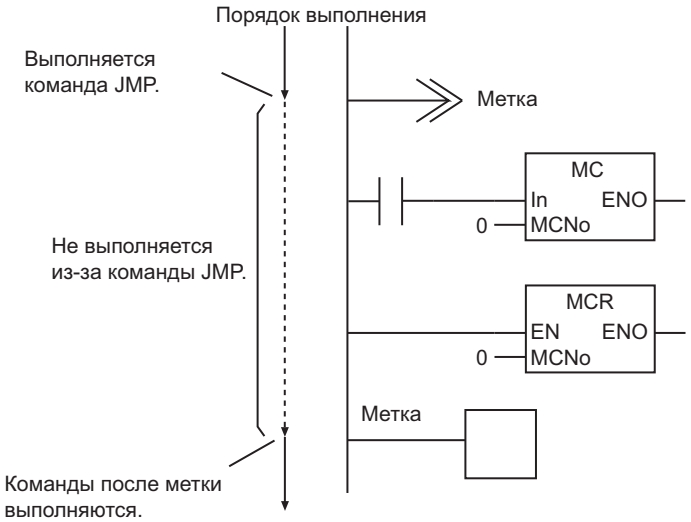


Пример программы представлен на рисунке ниже.
Если значение бита A = ЛОЖЬ, то область главного управления сброшена. Пока область главного управления сброшена, команда TON также находится в состоянии сброса. Команда MOVE не выполняется. Кроме того, команды Out и OutNot выдают состояние ЛОЖЬ в биты D и E.



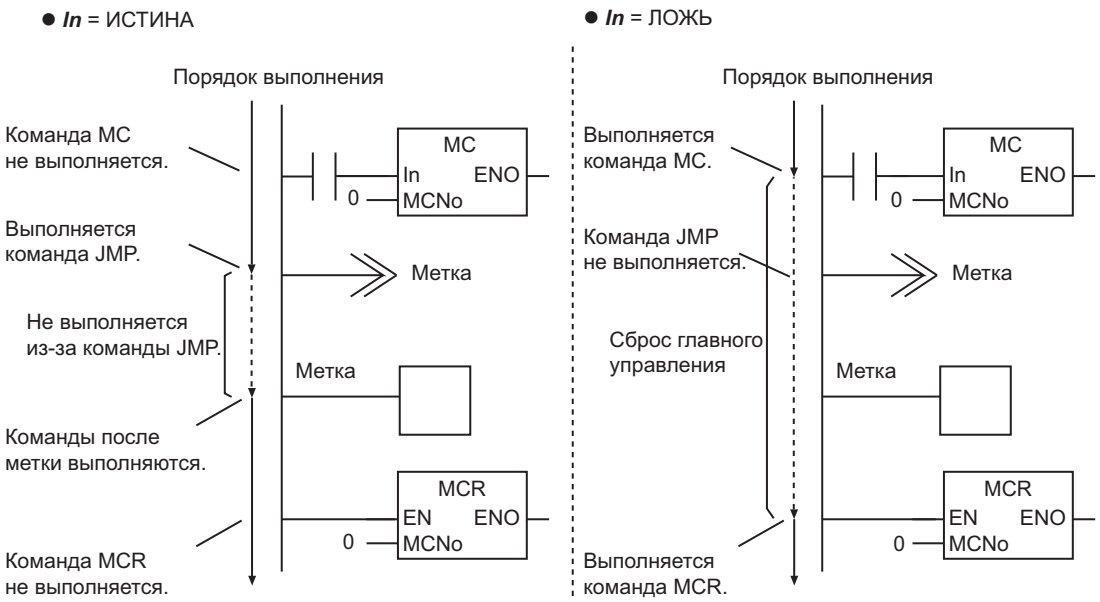
Меры предосторожности для обеспечения надлежащей эксплуатации

- Эти команды должны использоваться только в программе на языке релейно-контактных схем (LD). В программе на языке структурированного текста (ST) их использовать невозможно. Их также нельзя использовать внутри вставки на языке ST, используемой в программе на языке LD.
- Всегда подключайте вход *In* непосредственно к левой шине лестничной диаграммы. В параметр *In* невозможно передать переменную или константу.
- Всегда используйте команды MC и MCR в паре в пределах одного программного компонента.
- Всегда размещайте команду MCR после команды MC.
- Не вкладывайте пары команд MC и MCR друг в друга на глубину более 15 уровней.
- Вставка на языке ST, расположенная внутри области главного управления, не выполняется, когда область главного управления сброшена.
- Когда команды MC и MCR используются вместе с командой JMP, они работают следующим образом:
  a) На следующем рисунке показана пара «MC-MCR», расположенная внутри пары «JMP-Метка». В данном случае переход выполняется независимо от значения в *In*.

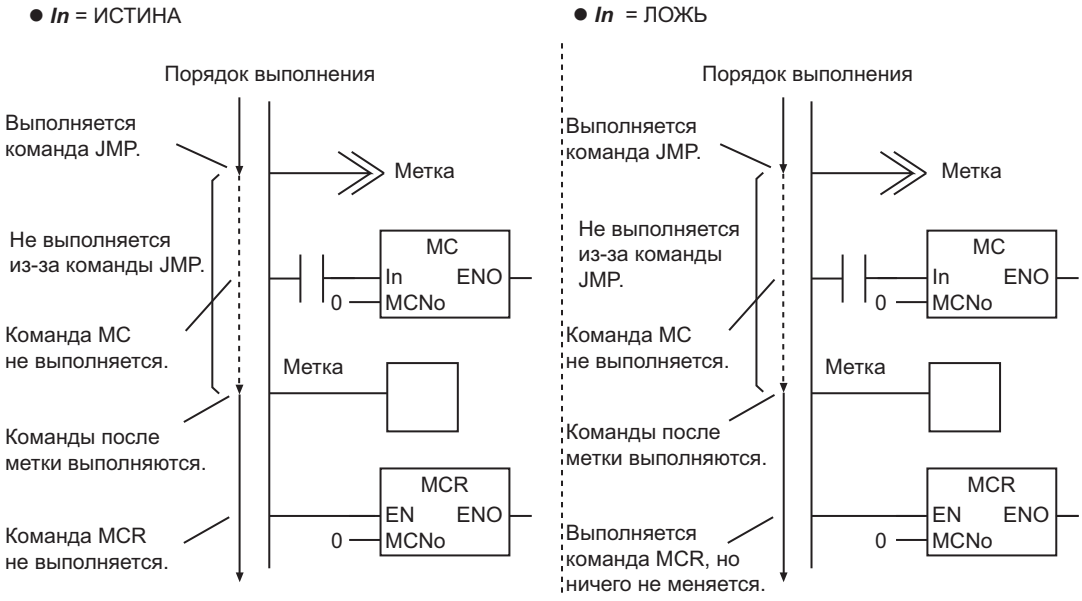


  b) На следующем рисунке показана пара «JMP-Метка», расположенная внутри пары «MC-MCR». Порядок работы, соответствующий данному случаю, поясняется в таблице ниже.

Значение в *In*	Работа
ИСТИНА	Область главного управления не сброшена. Выполняется переход.
ЛОЖЬ	Область главного управления сброшена. Переход не выполняется.

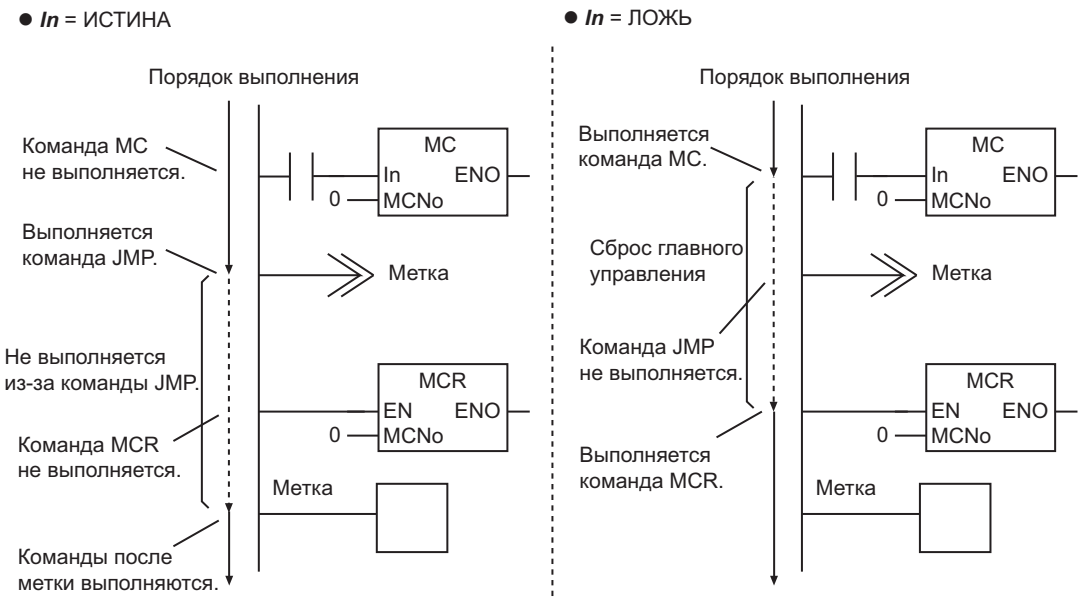


c) На приведенном ниже рисунке команды и метки расположены в следующем порядке: команда JMP, команда MC, метка, команда MCR. Сначала выполняется команда JMP. В результате команда MC не выполняется. Поэтому команды, находящиеся после метки, могут выполняться. Если значение *In* равно ЛОЖЬ, то команда MCR выполняется, но никаких изменений при этом не происходит.



d) На приведенном ниже рисунке команды и метки расположены в следующем порядке: команда MC, команда JMP, команда MCR и метка. Порядок работы, соответствующий данному случаю, поясняется в таблице ниже.

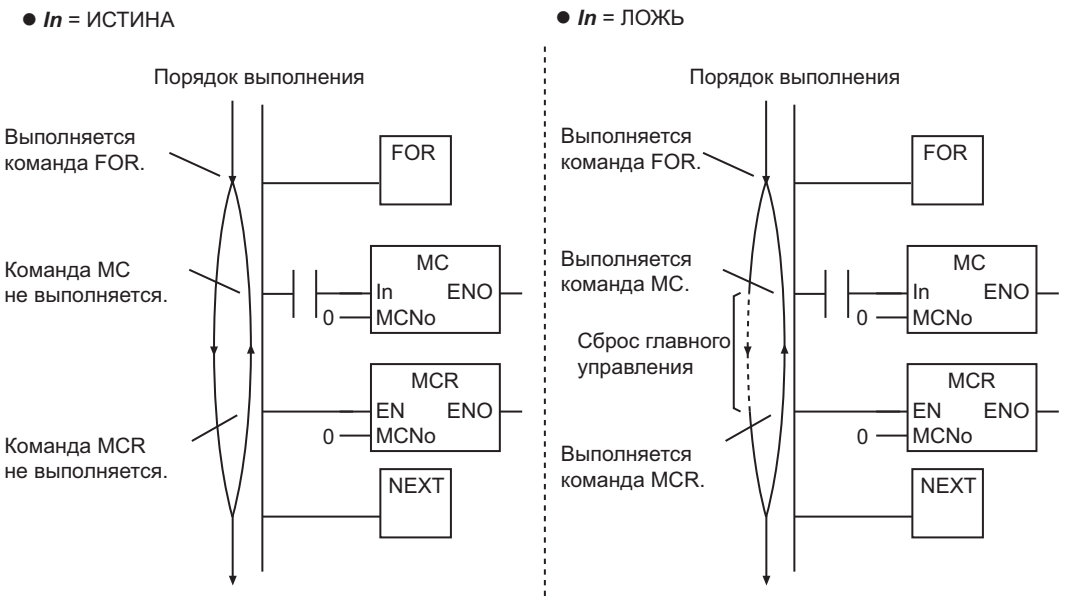
Значение в *In*	Работа
ИСТИНА	Область главного управления не сброшена. Выполняется переход.
ЛОЖЬ	Область главного управления сброшена. Переход не выполняется.



- Когда команды MC и MCR используются вместе с командами FOR и NEXT, они работают следующим образом:

  a) На следующем рисунке показана пара «MC-MCR», расположенная внутри пары «FOR-NEXT». Порядок работы, соответствующий данному случаю, поясняется в таблице ниже.

Значение в *In*	Работа
ИСТИНА	Область главного управления не сброшена. Выполняется цикл FOR.
ЛОЖЬ	Область главного управления сброшена. Цикл FOR выполняется, но команды, расположенные между командами MC и MCR, не выполняются.

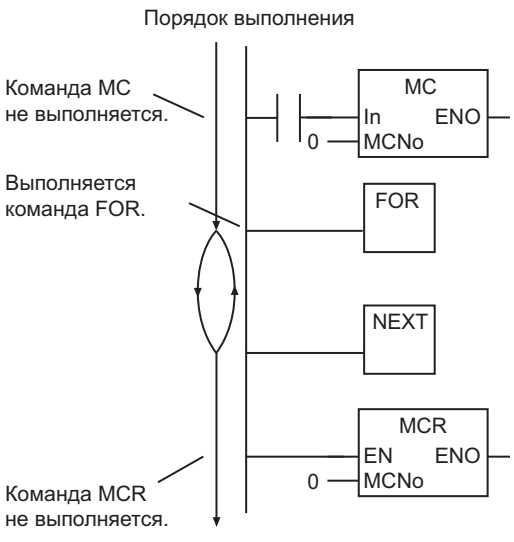


  b) На следующем рисунке показана пара «FOR-NEXT», расположенная внутри пары «MC-MCR». Порядок работы, соответствующий данному случаю, поясняется в таблице ниже.

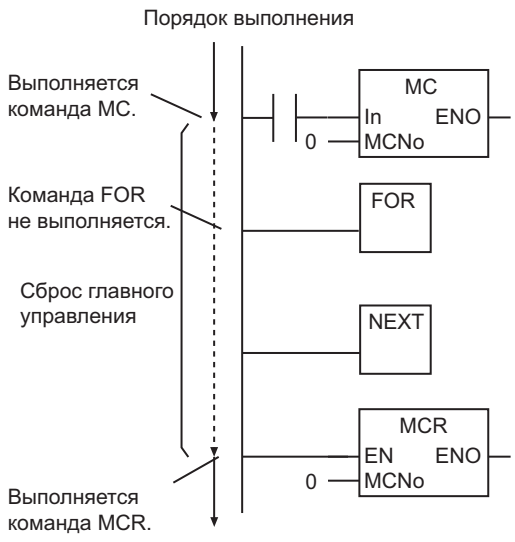
Значение в *In*	Работа
ИСТИНА	Область главного управления не сброшена. Выполняется цикл FOR.

Значение в *In*	Работа
ЛОЖЬ	Область главного управления сброшена. Цикл FOR не выполняется.

● ***In*** = ИСТИНА



● ***In*** = ЛОЖЬ



c) Если команды FOR, NEXT, MC и MCR будут расположены так, как показано ниже, произойдет ошибка сборки.
FOR, MC, NEXT, MCR или MC, FOR, MCR, NEXT

JMP

Команда JMP производит переход в указанную точку программы.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
JMP	Переход	FUN	Label	Нет

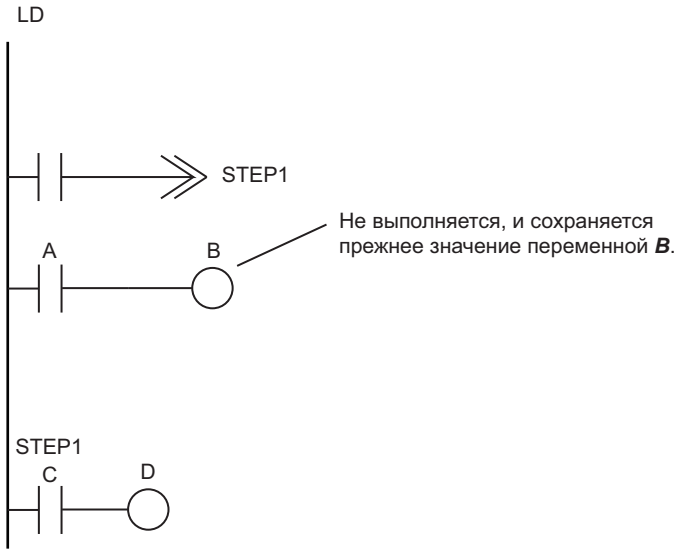
Переменные

Нет

Функция

Если условие выполнения = ИСТИНА, команда JMP передает управление в точку перехода, которая указывается в лестничной диаграмме с помощью метки (Label). В качестве метки можно ввести любую текстовую строку.

Пример программы представлен на рисунке ниже. В этом примере в качестве метки используется текстовая строка *STEP1*. Когда выполняется команда JMP, управление передается в точку, помеченную меткой *STEP1*. В данном примере команда Out, расположенная между командой JMP и меткой, не выполняется, и в переменной *B* сохраняется прежнее значение.



Дополнительная информация

- Переход может осуществляться в том числе и к команде, которая в данном сегменте программы расположена выше команды JMP.
- Одну и ту же метку можно использовать для нескольких команд JMP в программе. Другими словами, можно производить переход в одну и ту же точку программы из разных мест программы.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Опускать метки не допускается. Если какая-либо метка будет опущена, произойдет ошибка сборки.
- Команда JMP и соответствующая ей метка должны располагаться в пределах одного программного компонента и одного сегмента программы.
- В пределах одного сегмента программы не допускается размещать несколько одинаковых меток.
- Невозможно выполнить переход внутрь цикла FOR-NEXT из точки за пределами этого цикла.
- В следующей таблице перечислены ограничения на символы, которые могут использоваться в метках.

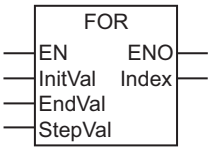
Параметр	Ограничения
Максимальное количество байтов	127 байт 127 символов при преобразовании в ACSII 31 символ при преобразовании в символы японского алфавита (включая однобайтовые знаки Кана)
Кодировка	UTF-8
Применимые символы	Не чувствительны к регистру. Английские буквенно-цифровые символы и символы других языков. Символы: _ (нижнее подчеркивание) и ~ (тильда)
Недопустимые текстовые строки	• Любая текстовая строка, начинающаяся с символа от 0 до 9 (коды символов от 16#30 до 16#39) ASCII. • Текстовая строка, состоящая только из одного символа _ (нижнее подчеркивание) ASCII. • Любая текстовая строка, содержащая подряд два и более символов _ (нижнее подчеркивание) ASCII. • Любая текстовая строка, начинающаяся с символа _ (нижнее подчеркивание) ASCII. • Любая текстовая строка, заканчивающаяся символом _ (нижнее подчеркивание) ASCII. • Любая текстовая строка, начинающаяся с *P_*.
Недопустимые символы	Пробел ! " # $ & ' ( ) * + , - . / : ; < = > ? @ [ ] ^ ` %

- В качестве меток не допускается использовать имена переменных.

FOR и NEXT

FOR : Обозначает начальную точку циклического выполнения и указывает условие повтора.
NEXT : Обозначает конечную точку циклического выполнения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST

FOR	Начало цикла	FUN		`FOR Index:=InitVal TO EndVal BY StepVal DO`
				`  выражение`
				`END_FOR*;`
				* В программе на языке ST для обозначения точки завершения циклического выполнения вместо ключевого слова *NEXT* используется ключевое слово *END_FOR*.

NEXT	Конец цикла	FUN		

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
InitVal	Начальное значение	Вход	Значение, записываемое в *Index* в начале цикла.	Зависит от типа данных.*1	---	*2
EndVal	Конечное значение		Значение *Index*, при котором цикл завершается.			
StepVal	Приращение		Значение, добавляемое к *Index* при каждой итерации цикла.	Зависит от типа данных.*3		*4
Index	Управляющая переменная	Выход	Счетчик цикла	Зависит от типа данных.	---	---

*1. При использовании команды в лестничной диаграмме задавайте значение *InitVal* так, чтобы оно было меньше значения *EndVal*.
*2. Если входной параметр будет опущен, значение по умолчанию применено не будет. Произойдет ошибка сборки.
*3. При использовании команды в лестничной диаграмме *0* и отрицательные числа не входят в диапазон допустимых значений. При использовании команды в программе на языке ST *0* не входит в диапазон допустимых значений.
*4. Если этот входной параметр будет опущен в лестничной диаграмме, значение по умолчанию применено не будет. Произойдет ошибка сборки. Если этот входной параметр будет опущен в программе на языке ST, будет применено значение по умолчанию *1*.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InitVal						OK	OK	OK	OK	OK	OK	OK	OK							
	Также можно указать перечисление, элемент массива или член структуры.*1																			
EndVal						OK	OK	OK	OK	OK	OK	OK	OK							
	Также можно указать элемент массива или член структуры.																			
StepVal						OK	OK	OK	OK	OK	OK	OK	OK							
	Также можно указать элемент массива или член структуры.																			
Index						OK	OK	OK	OK	OK	OK	OK	OK							
	Также можно указать элемент массива или член структуры.																			

*1. В лестничных диаграммах перечисления указывать нельзя.

Функция

Команды FOR и NEXT служат для организации циклического (т. е.повторяющегося) выполнения любых команд, размещенных между ними (в языке ST используются ключевые слова FOR и END_FOR).
Порядок работы цикла FOR-NEXT поясняется ниже.

1 Управляющая переменная *Index* принимает значение параметра *InitVal*.

2 Далее проверяется, соответствуют ли значения переменных *StepVal*, *Index* и *EndVal* условиям, приведенным в таблице ниже. Если условия соблюдаются, выполняется шаг 3. Если условия не соблюдаются, цикл не выполняется и управление передается следующей команде после команды NEXT (или END_FOR для ST).

Язык программирования	Условия для запуска цикла
Лестничная диаграмма (LD)	*StepVal* ≥ 0 и *Index* ≤ *EndVal*
Структурированный текст (ST)	*StepVal* ≥ 0 и *Index* ≤ *EndVal*
	StepVal < 0 и *Index* ≥ *EndVal*

3 Проверяется, соответствуют ли значения переменных *Index* и *EndVal* условиям, приведенным в таблице ниже. Если условия соблюдаются, выполняется шаг 4. Если условия не соблюдаются, цикл завершается и управление передается следующей команде после команды NEXT (или END_FOR для ST).

Язык программирования	Условия для продолжения цикла
Лестничная диаграмма (LD)	*Index* ≤ *EndVal*
Структурированный текст (ST)	Если *StepVal* ≥ 0, то *Index* должен быть ≤ *EndVal*
	Если *StepVal* < 0, то *Index* должен быть ≥ *EndVal*

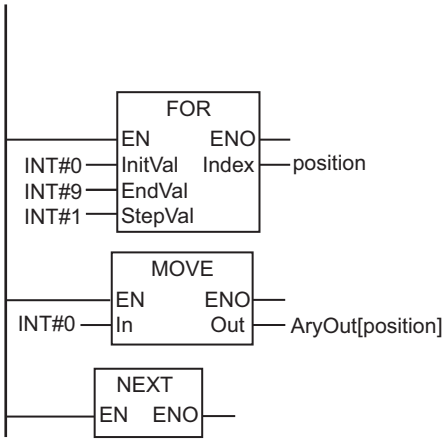
4 Выполняются один раз операции, заключенные между командами FOR и NEXT (или командой END_FOR для ST).

5 К значению *Index* добавляется значение параметра *StepVal*.

6 Процедура возвращается к шагу 3.

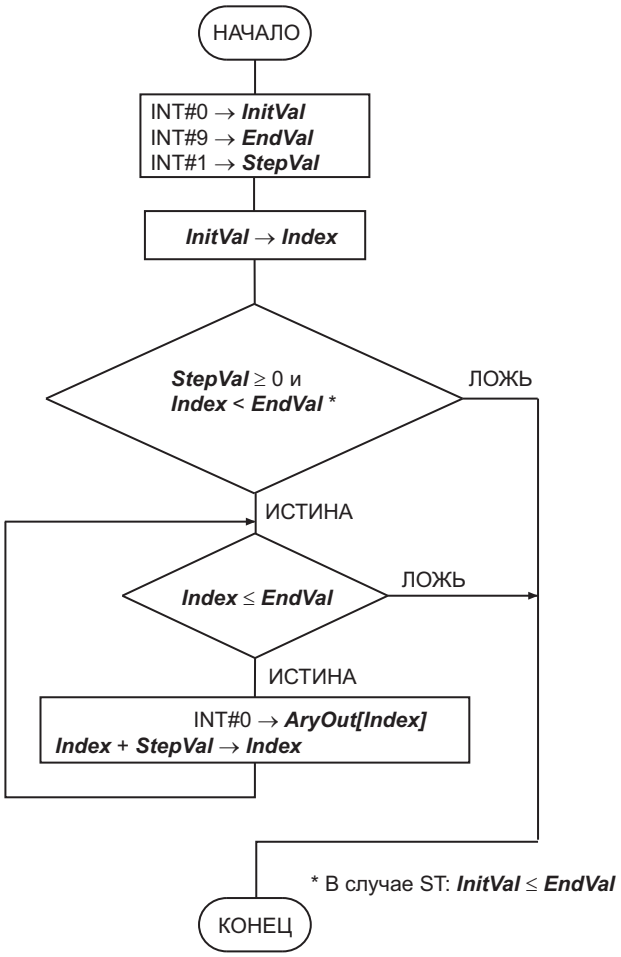
Ниже приведен пример для случая, когда *InitVal* = *INT#0*, *EndVal* = *INT#9*, а *StepVal* = *INT#1*. Команда MOVE выполняется 10 раз, и переменным массива AryOut[0]...AryOut[9] присваивается значение INT#0.

LD



ST

```
FOR position:=INT#0 TO INT#9 BY INT#1 DO
    AryOut[position]:=INT#0;
END_FOR;
```



Элементам ***AryOut[0]...AryOut[9]*** по очереди присваивается INT#0.

AryOut[0]	INT#0
AryOut[1]	INT#0
AryOut[2]	INT#0
AryOut[3]	INT#0
AryOut[4]	INT#0
AryOut[5]	INT#0
AryOut[6]	INT#0
AryOut[7]	INT#0
AryOut[8]	INT#0
AryOut[9]	INT#0

Пример программы на языке ST с использованием выражений или функций в качестве входных переменных

Если данные команды применяются в программе на языке ST, для входных переменных *InitVal*, *EndVal* и *StepVal* можно использовать одну из следующих форм записи:

- выражение с целочисленным результатом;
- функция, возвращающая целочисленное значение;
- функция, возвращающая перечислитель.

В представленном ниже примере в качестве *EndVal* используется функция, а в качестве *StepVal* — выражение.

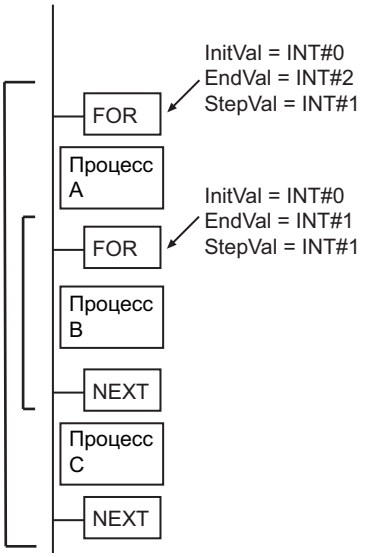
```
A:= DINT#1;
B:= DINT#2;
C:= REAL#9.6;
FOR i := 0 TO RoundUp(C) BY A+B DO
    DINTArray[i]:= i;
END_FOR;
```

Информация о версии

Использование выражений для параметров *EndVal* и *StepVal* возможно в Sysmac Studio версии 1.08 или более поздней версии.
Для параметра *InitVal* выражение можно использовать даже в Sysmac Studio версии 1.07 или более ранней версии.

Дополнительная информация

- Для отмены циклического выполнения используйте команду BREAK (или EXIT в программе на языке ST). Операции, расположенные между командами BREAK и NEXT, при этом выполнены не будут.
- Циклы FOR-NEXT (или циклы FOR-END_FOR для ST) можно вкладывать друг в друга. В приведенном ниже примере процессы выполняются в следующем порядке:
  Процесс A → Процесс B → Процесс B → Процесс C → Процесс A → Процесс B → Процесс B → Процесс C → Процесс A → Процесс B → Процесс B → Процесс C



Меры предосторожности для обеспечения надлежащей эксплуатации

- В лестничной диаграмме команды FOR и NEXT должны подключаться непосредственно к левой шине.
- Всегда используйте команды FOR и NEXT (инструкции FOR и END_FOR в языке ST) в паре. Если количество команд будет отличаться, произойдет ошибка программы.
- Парные команды FOR и NEXT должны располагаться в пределах одного сегмента программы.
- Тщательно продумайте и задайте условие для завершения циклического выполнения, чтобы не создать бесконечный цикл. При возникновении бесконечного цикла произойдет тайм-аут выполнения задачи.

  Если для входных параметров, передаваемых в переменные, используются значения, приведенные в следующей таблице, значение переменной *Index* никогда не превысит значения *EndVal*, поскольку значение переменной типа SINT не может быть больше 127. Следовательно, создастся бесконечный цикл.
  Никогда не используйте для параметра *EndVal* значение, являющееся максимальным для данного типа данных.

Переменная	Значение входного параметра
InitVal	SINT#0
EndVal	SINT#127
StepVal	SINT#1
Index	---

- В следующей таблице описывается порядок работы в зависимости от значений параметров *StepVal*, *InitVal* и *EndVal*.

Язык программирования	Значение *StepVal*	Значения *InitVal* и *EndVal*	Работа
Лестничная диаграмма (LD)	StepVal > 0	InitVal ≤ EndVal	Нормальная работа.
		InitVal > EndVal	Операции между командами FOR и NEXT не выполняются ни разу. К возникновению ошибки это не приводит.
	StepVal < 0	InitVal < EndVal	Операции между командами FOR и NEXT выполняются неопределенное количество раз. Не используйте такую комбинацию параметров. К возникновению ошибки это не приводит.
		InitVal ≥ EndVal	Операции между командами FOR и NEXT не выполняются ни разу. К возникновению ошибки это не приводит.
	StepVal = 0	InitVal < EndVal	Возникает бесконечный цикл, и происходит тайм-аут выполнения задачи.
		InitVal ≥ EndVal	Операции между командами FOR и NEXT не выполняются ни разу. К возникновению ошибки это не приводит.

Язык программирования	Значение *StepVal*	Значения *InitVal* и *EndVal*	Работа
Структурированный текст (ST)	StepVal > 0	InitVal ≤ EndVal	Нормальная работа.
		InitVal > EndVal	Операции между командами FOR и END_FOR не выполняются ни разу. К возникновению ошибки это не приводит.
	StepVal < 0	InitVal < EndVal	Операции между командами FOR и END_FOR не выполняются ни разу. К возникновению ошибки это не приводит.
		InitVal ≥ EndVal	Нормальная работа.
	StepVal = 0	InitVal ≤ EndVal	Возникает бесконечный цикл, и происходит тайм-аут выполнения задачи.
		InitVal > EndVal	Операции между командами FOR и END_FOR не выполняются ни разу. К возникновению ошибки это не приводит.

- Допускается вложение циклов FOR-NEXT на глубину до 15 уровней, однако при подсчете уровней вложения должны учитываться следующие команды: IF, CASE, FOR, WHILE и REPEAT.
- При наличии вложенных циклов на каждом уровне вложения необходимо использовать одну команду BREAK (или одну команду EXIT для языка ST), чтобы можно было отменить все циклы.
- Не используйте команды Jump (например, команду JMP) для прерывания циклического выполнения. Для отмены циклического выполнения всегда используйте команду BREAK (или EXIT в программе на языке ST).
- Изменение значений переменных *InitVal*, *EndVal* и *StepVal* во время работы цикла происходит по-разному в лестничной диаграмме и в программе на языке ST.

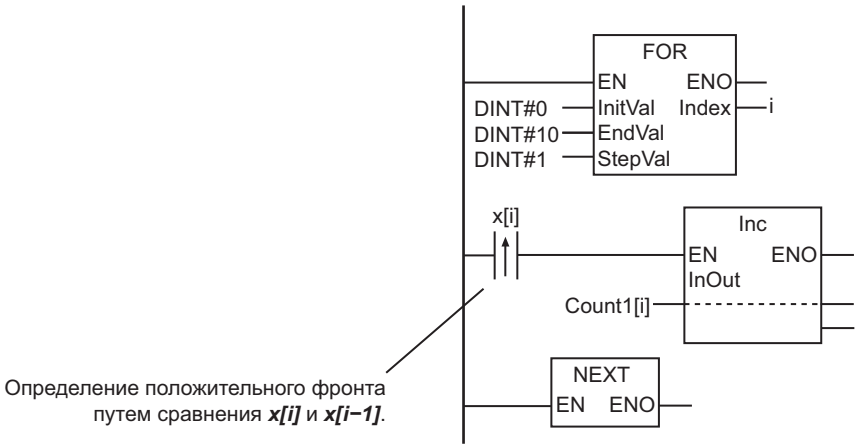
Переменная	Работа	
	Лестничная диаграмма (LD)	Структурированный текст (ST)
InitVal	Новое значение не применяется до завершения циклической обработки.	Новое значение не применяется до завершения циклической обработки.
EndVal	Новое значение применяется даже во время циклической обработки.	
StepVal	Программа может работать не так, как задумано. Не изменяйте значение этой переменной во время циклической обработки.	

- В лестничной диаграмме переменные *InitVal*, *EndVal*, *StepVal* и *Index* должны быть одного типа данных. В противном случае произойдет ошибка сборки.
- Выберите для переменной *Index* тип данных, который охватывает весь диапазон допустимых значений переменных *InitVal*, *EndVal* и *StepVal*. В противном случае произойдет ошибка сборки.
- В программе на языке LD значение переменной *Index* по завершении циклической обработки будет не таким, как в программе на языке ST. В лестничной диаграмме значение *StepVal* не добавляется к *Index* в конце циклической обработки. В программе на языке ST значение *StepVal* добавляется к *Index* в конце циклической обработки. Количество повторов в обоих языках одинаково.
  Ниже приведен пример для случая, когда *InitVal* = 1, *EndVal* = 100, а *StepVal* = 1.

Лестничная диаграмма (LD) : После 100 повторов значение переменной *Index* будет равно 100.
Структурированный текст (ST) : После 100 повторов значение переменной *Index* будет равно 101.

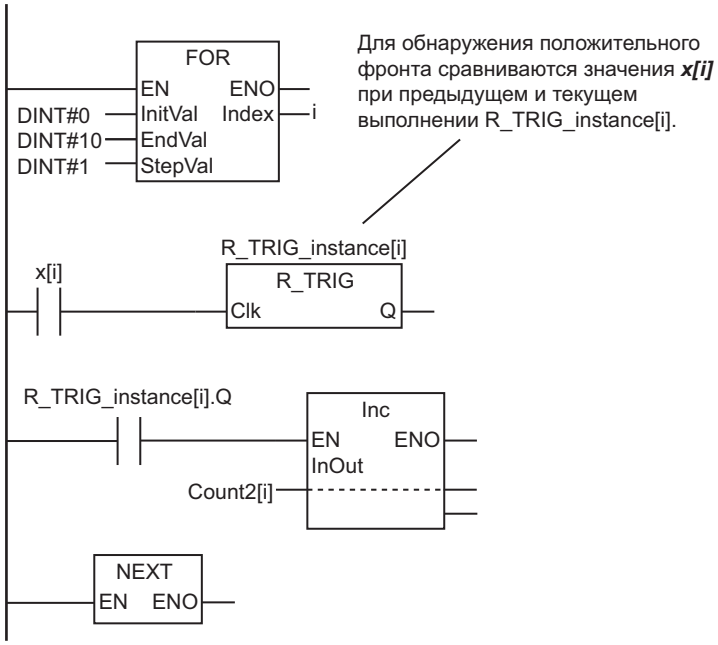
- Необходимо соблюдать осторожность, когда в лестничной диаграмме в теле цикла FOR используется команда LD, AND или OR, для которой указано различение положительного или отрицательного фронта, и при этом для данной команды используется массив.
  Положительный или отрицательный перепад (фронт) обнаруживается путем сравнения значения, содержавшегося в указанной переменной при предыдущем выполнении, со значением этой переменной при текущем выполнении. Обычно значение указанной переменной изменяется не каждый раз, когда выполняется команда. Но если в цикле FOR указан массив, элемент массива изменяется при каждом выполнении команды. Следовательно, при выявлении положительного или отрицательного фронта фактически сравниваются значения двух разных элементов массива.
  В следующей таблице показано, как соотношение значений x[i-1] и x[i] влияет на приращение значения переменной Count1[i].

Значение x[i-1]	Значение x[i]	Приращение Count1[i]
ИСТИНА	ИСТИНА	Не выполняется.
ИСТИНА	ЛОЖЬ	Не выполняется.
ЛОЖЬ	ИСТИНА	Выполняется.
ЛОЖЬ	ЛОЖЬ	Не выполняется.



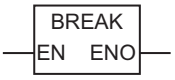
- В приведенной ниже программе команда R_TRIG обнаруживает положительный перепад переменной x[i]. Экземпляр команды R_TRIG предоставляется для каждого элемента x[i], что дает возможность обнаружить элементы x[i], в которых имеет место положительный перепад состояния. В следующей таблице отражена взаимосвязь между значением x[i] при предыдущем выполнении R_TRIG_instance[i], значением x[i] при текущем выполнении R_TRIG_instance[i] и выполнением приращения Count2[i].

Значение x[i] при предыдущем выполнении R_TRIG_instance[i]	Значение x[i] при текущем выполнении R_TRIG_instance[i]	Приращение Count2[i]
ИСТИНА	ИСТИНА	Не выполняется.
ИСТИНА	ЛОЖЬ	Не выполняется.
ЛОЖЬ	ИСТИНА	Выполняется.
ЛОЖЬ	ЛОЖЬ	Не выполняется.


FOR
EN
ENO
DINT#0
InitVal
Index
i
DINT#10
EndVal
DINT#1
StepVal
Для обнаружения положительного фронта сравниваются значения x[i] при предыдущем и текущем выполнении R_TRIG_instance[i].
R_TRIG_instance[i]
x[i]
R_TRIG
Clk
Q
R_TRIG_instance[i].Q
Inc
EN
ENO
InOut
Count2[i]
NEXT
EN
ENO

BREAK

Команда BREAK прекращает работу цикла, созданного самой внутренней командой FOR и следующей по порядку командой NEXT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
BREAK	Выход из цикла	FUN		Нет

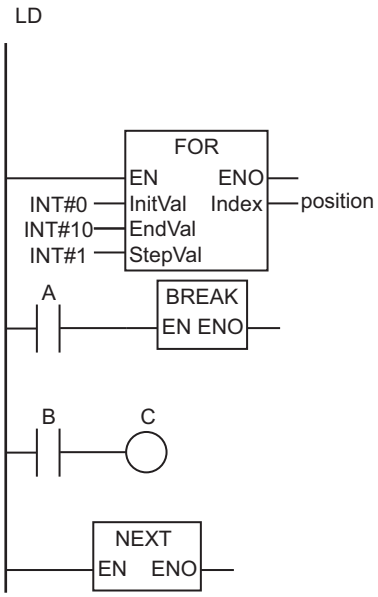
Переменные

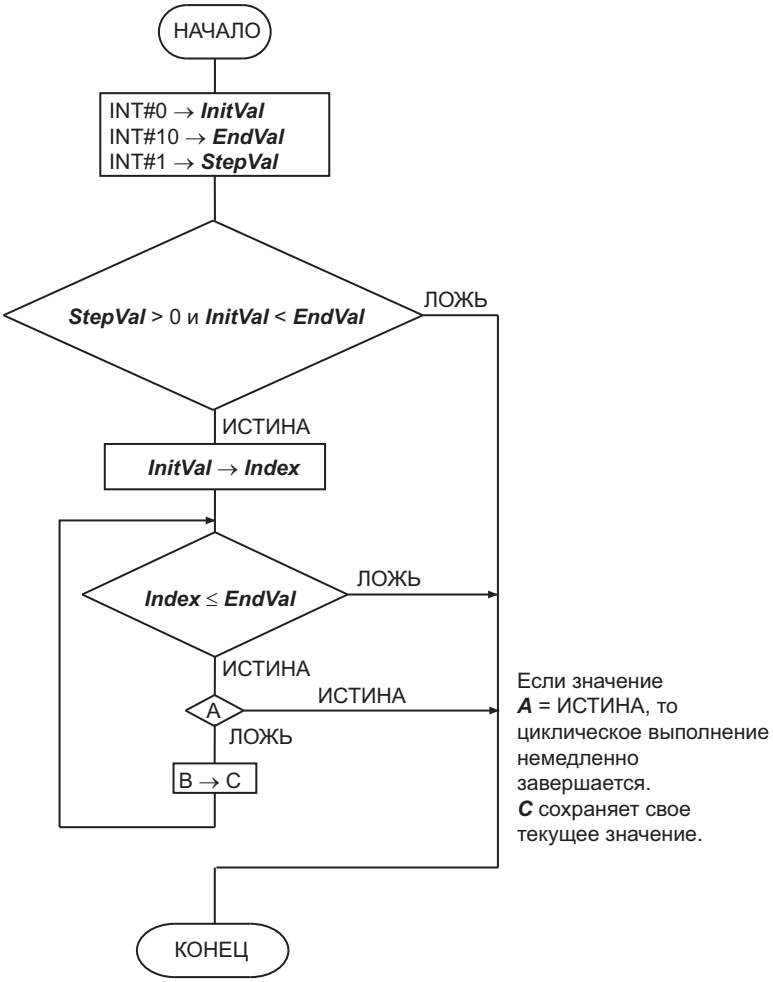
Нет

Функция

Команда BREAK прекращает работу наиболее глубоко вложенного цикла, т. е. цикла, заключенного между самой внутренней командой FOR и следующей по порядку командой NEXT. Она передает управление команде, которая следует по порядку за командой NEXT.
Операции, расположенные между командами BREAK и NEXT, при этом не выполняются.

Пример программы представлен на рисунке ниже. При каждой итерации цикла FOR проверяется значение переменной *A*. Если значение переменной *A* = ИСТИНА, то работа цикла немедленно завершается.
В данном примере не выполняется команда Out, расположенная после команды BREAK, в результате чего переменная *B* сохраняет свое прежнее значение.





Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда помещайте эту команду между командами FOR и NEXT.
- Если пары команд FOR и NEXT вкладываются друг в друга, необходимо предусмотреть одну команду BREAK на каждом уровне вложения, чтобы иметь возможность завершить выполнение всех циклов.
- Не используйте команды Jump (например, команду JMP) для прерывания циклического выполнения. Для отмены циклического выполнения следует использовать только команду BREAK.

Команды сравнения

Команда	Имя	Стр.
EQ (=)	Равно	стр. 2-108
NE (<>)	Не равно	стр. 2-111
LT (<), LE (<=), GT (>) и GE (>=)	Меньше/Меньше или равно/Больше/Больше или равно	стр. 2-114
EQascii	Сравнение текстовых строк: равно	стр. 2-118
NEascii	Сравнение текстовых строк: не равно	стр. 2-120
LTascii, LEascii, GTascii и GEascii	Сравнение текстовых строк: меньше/Сравнение текстовых строк: меньше или равно/Сравнение текстовых строк: больше/Сравнение текстовых строк: больше или равно	стр. 2-122
Cmp	Сравнение	стр. 2-125
ZoneCmp	Попадание в диапазон	стр. 2-128
TableCmp	Таблица сравнения	стр. 2-131
AryCmpEQ и AryCmpNE	Сравнение массивов: равно / Сравнение массивов: не равно	стр. 2-134
AryCmpLT, AryCmpLE, AryCmpGT и AryCmpGE	Сравнение массивов: меньше/Сравнение массивов: меньше или равно/ Сравнение массивов: больше/Сравнение массивов: больше или равно	стр. 2-137
AryCmpEQV и AryCmpNEV	Сравнение массива со значением: равно / Сравнение массива со значением: не равно	стр. 2-140
AryCmpLTV, AryCmpLEV, AryCmpGTV и AryCmpGEV	Сравнение массива со значением: меньше/Сравнение массива со значением: меньше или равно/Сравнение массива со значением: больше/Сравнение массива со значением: больше или равно	стр. 2-143

EQ (=)

Команда EQ (=) определяет, равны ли значения двух или большего числа переменных.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EQ(=)	Равно	FUN	(@)EQ EN In1 : InN Out (@)= EN In1 : InN Out	Out:=(In1=In2) & (In2=In3) & ⋯ & (InN-1=InN);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Сравниваемое значение	Вход	Значения для сравнения N = 2...5	Зависит от типа данных.	---	0*1
Out	Результат сравнения	Выход	Результат сравнения	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3, а входные параметры, которые подключаются к *In1* и *In2*, опущены, то применяются значения по умолчанию. Но если будет опущен входной параметр, который подключается к *In3*, то произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK *1	OK *1	OK *1	OK *1	OK *1
	Также допускается указывать перечисления.																			
Out	OK																			

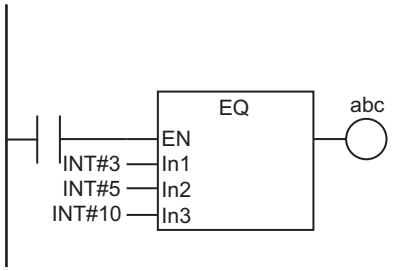
*1. В Sysmac Studio версии 1.02 или более поздней версии можно указывать значения следующих типов: TIME, DATE, TOD, DT и STRING. Если проект, созданный в Sysmac Studio версии 1.01 или более ранней версии, открывается в Sysmac Studio версии 1.02 или выше и в нем затем используются любые из этих типов данных, необходимо обновить отображаемую информацию. Для этого нужно щелкнуть правой кнопкой мыши команду на панели Edit (Редактирование) и выбрать **Update (Обновить)**. Если отображение команды обновлено не будет, произойдет ошибка сборки.

Функция

Команда EQ (=) определяет, равны ли друг другу значения от двух до пяти переменных *In1*...*InN*. Результат сравнения *Out* равен ИСТИНА, только если все значения равны. В противном случае *Out* содержит значение ЛОЖЬ.
При сравнении значений типа STRING "равенство" определяется, только если у текстовых строк совпадают и длина, и содержимое.

Ниже приведен пример, в котором *In1* = INT#3, *In2* = INT#5, а *In3* = INT#10. Значение переменной *abc* будет равно ЛОЖЬ.

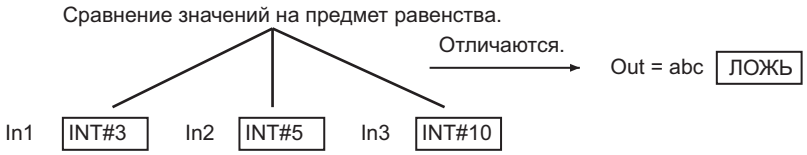
LD



ST

```
abc:=(INT#3=INT#5)&(INT#5=INT#10);
```

Команда EQ определяет, равны ли значения ***In1...In3***.
Если они отличаются, значение ***abc*** будет равно ЛОЖЬ.



Дополнительная информация

- Команды EQ и = работают абсолютно одинаково, отличаясь лишь формой записи. Используйте ту форму записи, которая вам более удобна.
- При сравнении значений типа TIME, DT или TOD необходимо определять их точность, чтобы сравнение выполнялось с такой же точностью. Для определения точности значений используйте следующие команды: *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 и *TruncTod* на стр. 2-757.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если типы данных значений *In1*...*InN* отличаются, они преобразуются в тип данных, который охватывает диапазоны значений всех используемых типов данных.
- Значения, являющиеся битовыми строками (BYTE, WORD, DWORD или LWORD), невозможно сравнивать с целочисленными значениями (SINT, INT, DINT, LINT, USINT, UINT, UDINT и ULINT). Значения, являющиеся битовыми строками, невозможно сравнивать со значениями вещественного типа (REAL и LREAL).
- Целые значения со знаком (SINT, INT, DINT и LINT) невозможно сравнивать с целыми значениями без знака (USINT, UINT, UDINT и ULINT).

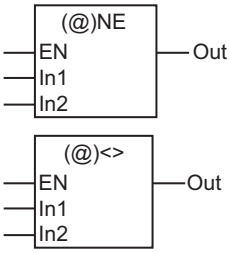
- В случае типов данных TIME, DATE, TOD, DT и STRING допускается сравнивать только значения одного типа. Если типы данных указанных для сравнения переменных будут отличаться, произойдет ошибка сборки.
- Перечисления можно сравнивать только с другими перечислениями. При сравнении перечислений типы данных также должны быть одинаковыми.
- Два значения, являющихся положительной бесконечностью, равно как и два значения, являющихся отрицательной бесконечностью, считаются равными.
- Если какой-либо из входов *In1*...*InN* содержит нечисловое значение, значение *Out* равно ЛОЖЬ.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.
- Если значения *In1*...*InN* являются вещественными числами, сравнение может быть выполнено неправильно из-за ошибки округления. Данную команду не следует применять для проверки равенства двух значений, если одно или несколько значений являются вещественными числами. Вместо этого следует использовать команду сравнения значений и определять, находится ли разница абсолютных значений двух чисел в пределах допустимого диапазона. Например, приведенный ниже фрагмент программы позволяет определить, равна ли сумма переменных *real_a* и *real_b* типа REAL числу 0,1. Два значения считаются равными, если переменная *boolv* типа BOOL равна ИСТИНА.

  boolv:=(ABS((real_a + real_b) - 0.1) < threshold);

  threshold: значение в пределах допустимого диапазона

NE (<>)

Команда NE (<>) определяет неравенство значений двух переменных.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NE(<>)	Не равно	FUN		Out:=(In1<>In2);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1 и In2	Сравниваемое значение	Вход	Значения для сравнения	Зависит от типа данных.	---	*1
Out	Результат сравнения	Выход	Результат сравнения	Зависит от типа данных.	---	---

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1 и In2	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК *1	ОК *1	ОК *1	ОК *1	ОК *1
	Также допускается указывать перечисления.																			
Out	ОК																			

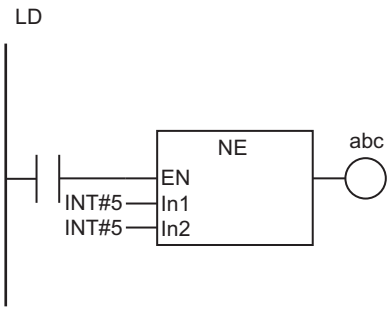
*1. В Sysmac Studio версии 1.02 или более поздней версии можно указывать значения следующих типов: TIME, DATE, TOD, DT и STRING. Если проект, созданный в Sysmac Studio версии 1.01 или более ранней версии, открывается в Sysmac Studio версии 1.02 или выше и в нем затем используются любые из этих типов данных, необходимо обновить отображаемую информацию. Для этого нужно щелкнуть правой кнопкой мыши команду на панели Edit (Редактирование) и выбрать **Update (Обновить)**. Если отображение команды обновлено не будет, произойдет ошибка сборки.

Функция

Команда NE (<>) определяет, отличается ли содержимое переменной *In1* от содержимого переменной *In2*.

При обнаружении отличия результат сравнения *Out* равен ИСТИНА. Если переменные не отличаются, результат равен ЛОЖЬ.
При сравнении значений типа STRING "равенство" определяется, только если у текстовых строк совпадают и длина, и содержимое.

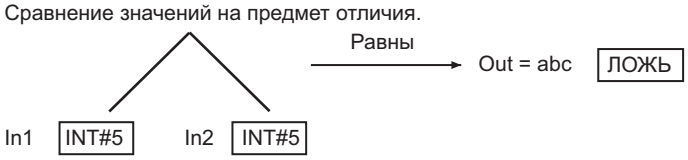
Ниже приведен пример для случая, когда *In1* = *In2* (оба входа содержат значение INT#5). Значение переменной *abc* будет равно ЛОЖЬ.



ST

```
abc:=(INT#5<>INT#5);
```

Команда NE определяет, отличаются ли ***In1*** и ***In2***.
Если они одинаковы, значение ***abc*** будет равно ЛОЖЬ.



Дополнительная информация

- Команды NE и <> работают абсолютно одинаково, отличаясь лишь формой записи. Используйте ту форму записи, которая вам более удобна.
- При сравнении значений типа TIME, DT или TOD необходимо определять их точность, чтобы сравнение выполнялось с такой же точностью. Для определения точности значений используйте следующие команды: *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 и *TruncTod* на стр. 2-757.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если типы данных значений *In1* и *In2* отличаются, тип данных с меньшим диапазоном допустимых значений приводится к типу данных, который поддерживает диапазоны значений обоих типов данных.
- Значения, являющиеся битовыми строками (BYTE, WORD, DWORD или LWORD), невозможно сравнивать с целочисленными значениями (SINT, INT, DINT, LINT, USINT, UDINT, ULINT). Значения, являющиеся битовыми строками, невозможно сравнивать со значениями вещественного типа (REAL и LREAL).
- Целые значения со знаком (SINT, INT, DINT и LINT) невозможно сравнивать с целыми значениями без знака (USINT, UINT, UDINT и ULINT).
- В случае типов данных TIME, DATE, TOD, DT и STRING допускается сравнивать только значения одного типа. Если типы данных указанных для сравнения переменных будут отличаться, произойдет ошибка сборки.

- Перечисления можно сравнивать только с другими перечислениями. При сравнении перечислений типы данных также должны быть одинаковыми.
- Два значения, являющихся положительной бесконечностью, равно как и два значения, являющихся отрицательной бесконечностью, считаются равными.
- Если какой-либо из входов, *In1* или *In2*, содержит нечисловое значение, значение *Out* равно ИСТИНА.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.
- Если значения *In1*и *In2* являются вещественными числами, сравнение может быть выполнено неправильно из-за ошибки округления. Данную команду не следует применять для проверки неравенства двух значений, если одно или оба значения являются вещественными числами. Вместо этого следует использовать команду сравнения значений и определять, выходит ли разница абсолютных значений двух чисел за пределы допустимого диапазона. Например, приведенный ниже фрагмент программы позволяет определить, отличается ли сумма переменных *real_a* и *real_b* типа REAL от числа 0,1. Два значения считаются не равными, если переменная *boolv* типа BOOL равна ИСТИНА.

  boolv:= (ABS((real_a + real_b) - 0.1) > threshold);

  threshold: значение в пределах допустимого диапазона

LT (<), LE (<=), GT (>) и GE (>=)

Эти команды позволяют сравнить два или больше значений и определить, как они соотносятся между собой.

LT (<) : Выполняет сравнение «меньше».
LE (<=) : Выполняет сравнение «меньше или равно».
GT (>) : Выполняет сравнение «больше».
GE (>=) : Выполняет сравнение «больше или равно».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LT(<)	Меньше	FUN	(@)LT EN In1 : InN Out (@)< EN In1 : InN Out	Out:=(In1<In2) & (In2<In3) & ⋯ & (InN-1<InN);
LE(<=)	Меньше или равно	FUN	(@)LE EN In1 : InN Out (@)<= EN In1 : InN Out	Out:=(In1<=In2) & (In2<=In3) & ⋯ & (InN-1<=InN);
GT(>)	Больше	FUN	(@)GT EN In1 : InN Out (@)> EN In1 : InN Out	Out:=(In1>In2) & (In2>In3) & ⋯ & (InN-1>InN);
GE(>=)	Больше или равно	FUN	(@)GE EN In1 : InN Out (@)>= EN In1 : InN Out	Out:=(In1>=In2) & (In2>=In3) & ⋯ & (InN-1>=InN);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Сравниваемое значение	Вход	Значения для сравнения N = 2...5	Зависит от типа данных.	---	*1
Out	Результат сравнения	Выход	Результат сравнения	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3, а входные параметры, которые подключаются к *In1* и *In2*, опущены, то применяются значения по умолчанию. Но если будет опущен входной параметр, который подключается к *In3*, то произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN		OK *1	OK *1	OK *1	OK *1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK *1	OK *1	OK *1	OK *1	OK *1
Out	OK																			

*1. В Sysmac Studio версии 1.02 или более поздней версии можно указывать значения следующих типов: BYTE, WORD, DWORD, LWORD, TIME, DATE, TOD, DT и STRING. Если проект, созданный в Sysmac Studio версии 1.01 или более ранней версии, открывается в Sysmac Studio версии 1.02 или выше и в нем затем используются любые из этих типов данных, необходимо обновить отображаемую информацию. Для этого нужно щелкнуть правой кнопкой мыши команду на панели Edit (Редактирование) и выбрать **Update (Обновить)**. Если отображение команды обновлено не будет, произойдет ошибка сборки.

Функция

Эти команды сравнивают значения на входах *In1*...*InN* (N = 2...5).
Выходное значение *Out* для каждой команды представлено в таблице ниже.

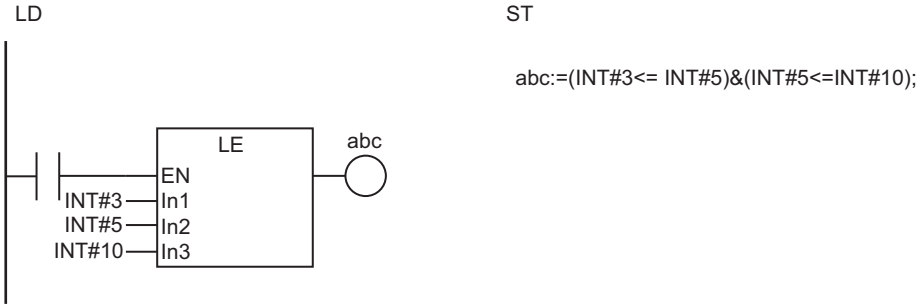
Команда	Значение *Out*
LT (<)	Если *In1* < *In2* < ... < *InN*, то *Out* = ИСТИНА. Иначе — ЛОЖЬ.
LE (<=)	Если *In1* <= *In2* <= ... <= *InN*, то *Out* = ИСТИНА. Иначе — ЛОЖЬ.
GT (>)	Если *In1* > *In2* > ... > *InN*, то *Out* = ИСТИНА. Иначе — ЛОЖЬ.
GE (>=)	Если *In1* >= *In2* >= ... >= *InN*, то *Out* = ИСТИНА. Иначе — ЛОЖЬ.

Как соотносятся между собой значения, не являющиеся целыми или вещественными числами, поясняется в таблице ниже.

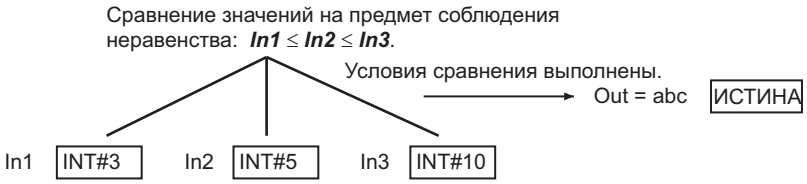
Тип данных	Соотношение
BYTE, WORD, DWORD или LWORD	Значения сравниваются как целые числа без знака.
TIME	За большее принимается значение, которое больше в числовом выражении.
DATE, TOD или DT	За большее принимается более поздняя дата или время суток.

Тип данных	Соотношение
STRING	Применяются те же правила, что и для команд *LTascii, LEascii, GTascii u GEascii* на стр. 2-122. Подробную информацию см. на указанной странице.

Ниже показан пример использования команды LE для случая, когда *In1* = INT#3, *In2* = INT#5, а *In3* = INT#10. Значение переменной *abc* будет равно ИСТИНА.



Команда LE определяет, выполняется ли условие: ***In1*** ≤ ***In2*** ≤ ***In3***.
Если условия сравнения соблюдаются, значение ***abc*** будет равно ИСТИНА.



Дополнительная информация

- Команды LT и <, команды LE и <=, команды GT и > и команды GE и >= работают абсолютно одинаково, отличаясь лишь формой записи. Используйте ту форму записи, которая вам более удобна.
- При сравнении значений типа TIME, DT или TOD необходимо определять их точность, чтобы сравнение выполнялось с такой же точностью. Для определения точности значений можно использовать следующие команды: *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 и *TruncTod* на стр. 2-757.

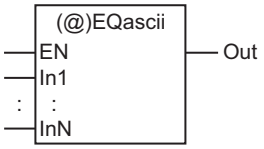
Меры предосторожности для обеспечения надлежащей эксплуатации

- Если типы данных значений *In1*...*InN* отличаются, перед сравнением они будут приведены к типу данных, который охватывает любые возможные значения каждого из используемых типов данных.
- Целые значения со знаком (SINT, INT, DINT и LINT) невозможно сравнивать с целыми значениями без знака (USINT, UINT, UDINT и ULINT).
- Значения, являющиеся битовыми строками (BYTE, WORD, DWORD или LWORD), невозможно сравнивать с целочисленными значениями (SINT, INT, DINT, LINT, USINT, UINT, UDINT или ULINT). Значения, являющиеся битовыми строками, невозможно сравнивать со значениями вещественного типа (REAL или LREAL).

- В случае типов данных TIME, DATE, TOD, DT и STRING допускается сравнивать только значения одного типа. Если типы данных указанных для сравнения переменных будут отличаться, произойдет ошибка сборки.
- Если *In1*...*InN* являются вещественными числами и содержат какое-либо десятичное число с бесконечной дробной частью, может произойти ошибка и результат сравнения может быть неправильным.
- Два значения, являющихся положительной бесконечностью, равно как и два значения, являющихся отрицательной бесконечностью, считаются равными.
- Если какой-либо из входов *In1*...*InN* содержит нечисловое значение, значение *Out* равно ЛОЖЬ.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

EQascii

Команда EQascii определяет, совпадают ли друг с другом две или более текстовых строк.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EQascii	Сравнение текстовых строк: равно	FUN		Out:=EQascii(In1, ··, InN);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Сравниваемые текстовые строки	Вход	Текстовые строки для сравнения N = 2...5	Зависит от типа данных.	---	""*1
Out	Результат сравнения	Выход	Результат сравнения	Зависит от типа данных.	---	---

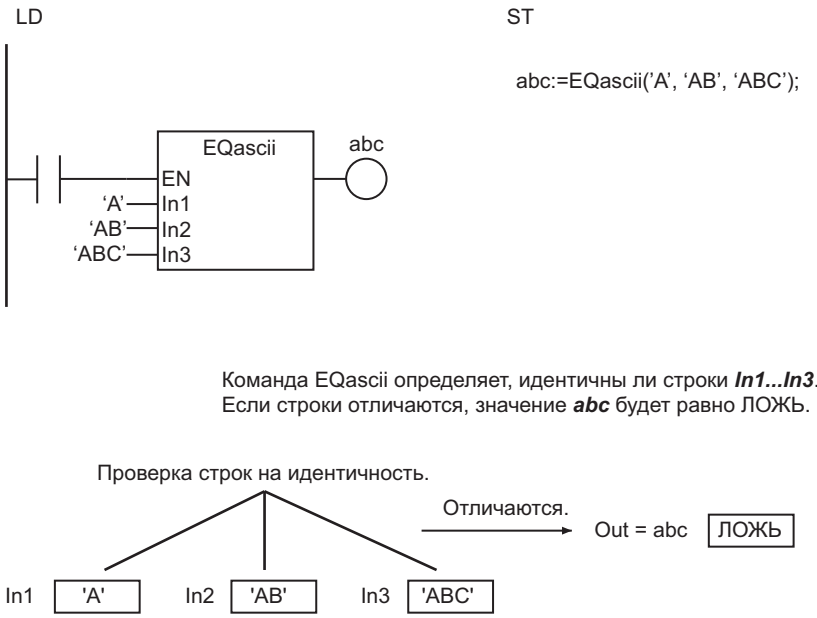
*1. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3, а входные параметры, которые подключаются к *In1* и *In2*, опущены, то применяются значения по умолчанию. Но если будет опущен входной параметр, который подключается к *In3*, то произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN																				OK
Out	OK																			

Функция

Команда EQascii позволяет определить, совпадают ли друг с другом от двух до пяти текстовых строк *In1*...*InN*.
Если все строки совпадают, результат сравнения *Out* становится равен ИСТИНА. В противном случае *Out* содержит значение ЛОЖЬ.
Текстовые строки считаются "совпадающими (равными)", если они имеют одинаковую длину и одинаковое содержимое.

Ниже приведен пример, в котором *In1* = «A», *In2* = «AB», а *In3* = «ABC». Значение переменной *abc* будет равно ЛОЖЬ.



Дополнительная информация

Командами сравнения текстовых строк удобно пользоваться, когда требуется изменить порядок текстовых строк в соответствии с кодами символов. Например, коды символов букв алфавита располагаются в том же порядке, что и буквы алфавита. Это позволяет упорядочивать текстовые строки по алфавиту.

Информация о версии

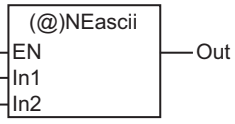
В Sysmac Studio версии 1.02 или более поздней версии для сравнения текстовых строк также можно использовать команду *EQ (=)* на стр. 2-108. Использование команды EQ (=) для сравнения текстовых строк не отличается от использования команды EQascii.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Не используйте эту команду в качестве самой правой команды в цепи. В противном случае в Sysmac Studio возникнет ошибка и загрузить программу пользователя в контроллер будет невозможно.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.
- Текстовые строки, указываемые для входов *In1*...*InN*, должны содержать только символы ASCII.

NEascii

Команда NEascii определяет, отличаются ли друг от друга две текстовые строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NEascii	Сравнение текстовых строк: не равно	FUN		Out:=NEascii(In1, In2);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1 и In2	Сравниваемые текстовые строки	Вход	Текстовые строки для сравнения	Зависит от типа данных.	---	*1
Out	Результат сравнения	Выход	Результат сравнения	Зависит от типа данных.	---	---

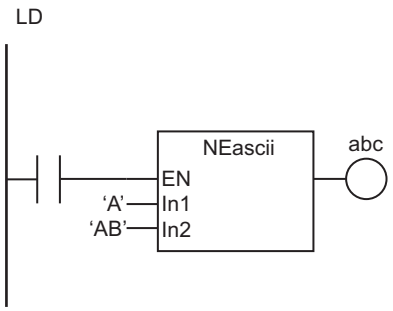
*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1 и In2																				OK
Out	OK																			

Функция

Команда NEascii определяет, отличаются ли друг от друга две текстовые строки *In1* и *In2*.
Если они отличаются, результат сравнения *Out* будет равен ИСТИНА, если совпадают — ЛОЖЬ.
Текстовые строки считаются "совпадающими (равными)", если они имеют одинаковую длину и одинаковое содержимое.

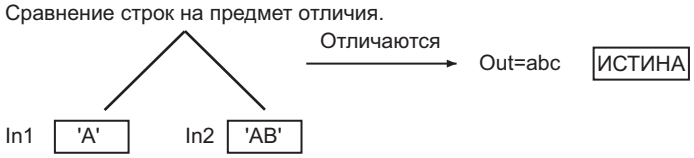
Ниже приведен пример, в котором *In1* = 'A', а *In2* = 'AB'. Значение переменной *abc* будет равно ИСТИНА.



ST

```
abc:=NEascii('A', 'AB');
```

Команда NEascii определяет, отличаются ли ***In1*** и ***In2***.
Если они отличаются, значение ***abc*** будет равно ИСТИНА.



Дополнительная информация

Командами сравнения текстовых строк удобно пользоваться, когда требуется изменить порядок текстовых строк в соответствии с кодами символов. Например, коды символов букв алфавита располагаются в том же порядке, что и буквы алфавита. Это позволяет упорядочивать текстовые строки по алфавиту.

Информация о версии

В Sysmac Studio версии 1.02 или более поздней версии для сравнения текстовых строк также можно использовать команду *NE (<>)* на стр. 2-111. Использование команды NE (<>) для сравнения текстовых строк не отличается от использования команды NEascii.

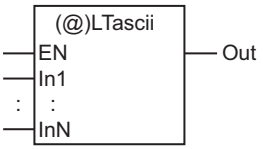
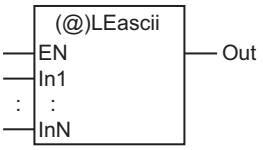
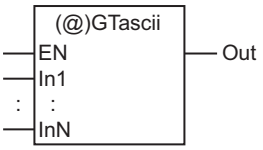
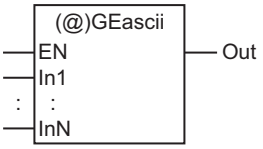
Меры предосторожности для обеспечения надлежащей эксплуатации

- Не используйте эту команду в качестве самой правой команды в цепи. В противном случае в Sysmac Studio возникнет ошибка и загрузить программу пользователя в контроллер будет невозможно.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.
- Текстовые строки, указываемые для входов *In1* и *In2*, должны содержать только символы ASCII.

LTascii, LEascii, GTascii и GEascii

Эти команды сравнивают коды символов двух или более текстовых строк.

LTascii : Выполняет сравнение «меньше».
LEascii : Выполняет сравнение «меньше или равно».
GTascii : Выполняет сравнение «больше».
GEascii : Выполняет сравнение «больше или равно».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LTascii	Сравнение текстовых строк: меньше	FUN		Out:=LTascii(In1, ⋯, InN);
LEascii	Сравнение текстовых строк: меньше или равно	FUN		Out:=LEascii(In1, ⋯, InN);
GTascii	Сравнение текстовых строк: больше	FUN		Out:=GTascii(In1, ⋯, InN);
GEascii	Сравнение текстовых строк: больше или равно	FUN		Out:=GEascii(In1, ⋯, InN);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Сравниваемые текстовые строки	Вход	Текстовые строки для сравнения N = 2...5	Зависит от типа данных.	---	''*1
Out	Результат сравнения	Выход	Результат сравнения	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3, а входные параметры, которые подключаются к *In1* и *In2*, опущены, то применяются значения по умолчанию. Но если будет опущен входной параметр, который подключается к *In3*, то произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN																				OK
Out	OK																			

Функция

Эти команды сравнивают коды символов от двух до пяти текстовых строк в *In1*...*InN* (N = 2...5). Выходное значение *Out* для каждой команды представлено в таблице ниже.

Команда	Значение *Out*
LTascii	Если *In1* < *In2* < ... < *InN*, то *Out* = ИСТИНА. Иначе — ЛОЖЬ.
LEascii	Если *In1* ≤ *In2* ≤ ... ≤ *InN*, то *Out* = ИСТИНА. Иначе — ЛОЖЬ.
GTascii	Если *In1* > *In2* > ... > *InN*, то *Out* = ИСТИНА. Иначе — ЛОЖЬ.
GEascii	Если *In1* ≥ *In2* ≥ ... ≥ *InN*, то *Out* = ИСТИНА. Иначе — ЛОЖЬ.

Сравниваются значения кодов символов. Процедура сравнения описана ниже:
Сначала сравниваются коды первых символов во всех текстовых строках. Если коды символов отличаются, результат сравнения текстовых строк определяется соотношением значений кодов этих символов.
Если коды символов совпадают, выполняется сравнение для следующих по порядку символов. Процедура продолжается, пока не будут обнаружены отличающиеся коды символов.
Если текстовые строки имеют разную длину, для завершения сравнения к более короткой текстовой строке добавляются символы NULL (16#00).

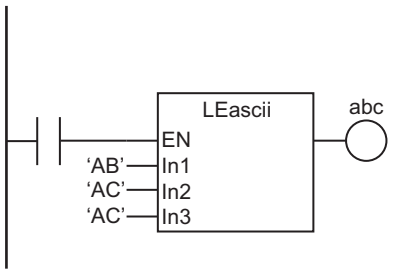
Ниже показано, как определяется соотношение между разными строками:

'AD'(16#414400) < 'BC'(16#424400)
'ADC' (16#41444300) < 'B' (16#42000000)
'ABC' (16#41424300) < 'ABD' (16#41424400)
'ABC' (16#41424300) > 'AB' (16#41420000)
'AB' (16#414200) = 'AB' (16#414200)

Если текстовая строка содержит многобайтовые символы, перед сравнением символы разделяются на отдельные байты. Например, двухбайтовый символ 16#C281 обрабатывается как два байта: 16#C2 и 16#81.

Ниже показан пример использования команды LEascii для случая, когда *In1* = "AB", *In2* = "AC", а *In3* = "AC". Значение переменной *abc* будет равно ИСТИНА.

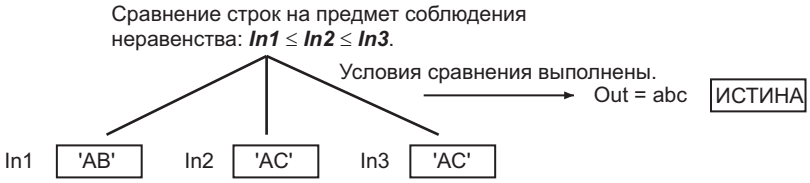
LD



ST

```
abc:=LEascii('AB', 'AC', 'AC');
```

Команда LEascii определяет, выполняется ли условие: ***In1*** ≤ ***In2*** ≤ ***In3***.
Если условия сравнения соблюдаются, ***abc*** будет равно ИСТИНА.



Дополнительная информация

Командами сравнения текстовых строк удобно пользоваться, когда требуется изменить порядок текстовых строк в соответствии с кодами символов. Например, коды символов букв алфавита располагаются в том же порядке, что и буквы алфавита. Это позволяет упорядочивать текстовые строки по алфавиту.

Информация о версии

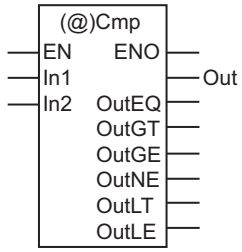
В Sysmac Studio версии 1.02 или более поздней версии для сравнения текстовых строк также можно использовать команды *LT (<), LE (<=), GT (>) и GE (>=)* на стр. 2-114. Использование команд LT (<), LE (<=), GT (>) и GE (>=) для сравнения текстовых строк не отличается от использования команд LTascii, LEascii, GTascii и GEascii.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Не используйте эту команду в качестве самой правой команды в цепи. В противном случае в Sysmac Studio возникнет ошибка и загрузить программу пользователя в контроллер будет невозможно.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.
- Текстовые строки, указываемые для входов *In1*...*InN*, должны содержать только символы ASCII.

Cmp

Команда Cmp сравнивает два значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Cmp	Сравнение	FUN		Out:=Cmp(In1, In2, OutEQ, OutGT, OutGE, OutNE, OutLT, OutLE); Выход Out может быть опущен.

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1 и In2	Сравниваемое значение	Вход	Значения для сравнения	Зависит от типа данных.	---	*1
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---
OutEQ	Флаг «Равно»		Флаг «Равно»	Зависит от типа данных.		
OutGT	Флаг «Больше»		Флаг «Больше»			
OutGE	Флаг «Больше или равно»		Флаг «Больше или равно»			
OutNE	Флаг «Не равно»		Флаг «Не равно»			
OutLT	Флаг «Меньше»		Флаг «Меньше»			
OutLE	Флаг «Меньше или равно»		Флаг «Меньше или равно»			

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1 и In2						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
Out	OK																			
OutEQ	OK																			
OutGT	OK																			
OutGE	OK																			
OutNE	OK																			
OutLT	OK																			
OutLE	OK																			

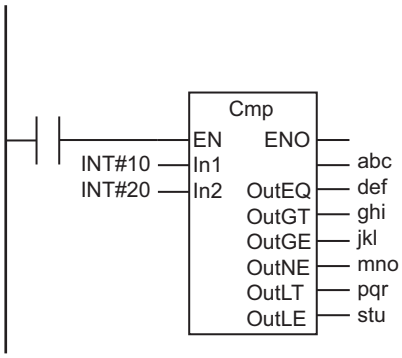
Функция

Команда Cmp сравнивает два значения (*In1* и *In2*) и соответствующим образом изменяет состояния выходных флагов.
Флаги принимают значения, указанные ниже:

Флаг	Значение
OutEQ	Если *In1* равно *In2*, флаг показывает ИСТИНА. В противном случае флаг показывает ЛОЖЬ.
OutGT	Если *In1* больше *In2*, флаг показывает ИСТИНА. В противном случае флаг показывает ЛОЖЬ.
OutGE	Если *In1* больше или равно *In2*, флаг показывает ИСТИНА. В противном случае флаг показывает ЛОЖЬ.
OutNE	Если *In1* не равно *In2*, флаг показывает ИСТИНА. В противном случае флаг показывает ЛОЖЬ.
OutLT	Если *In1* меньше *In2*, флаг показывает ИСТИНА. В противном случае флаг показывает ЛОЖЬ.
OutLE	Если *In1* меньше или равно *In2*, флаг показывает ИСТИНА. В противном случае флаг показывает ЛОЖЬ.

Ниже приведен пример, в котором *In1* = INT#10, а *In2* = INT#20. Переменные *def*, *ghi* и *jkl* будут равны ЛОЖЬ, а переменные *abc*, *mno*, *pqr* и *stu* будут равны ИСТИНА.

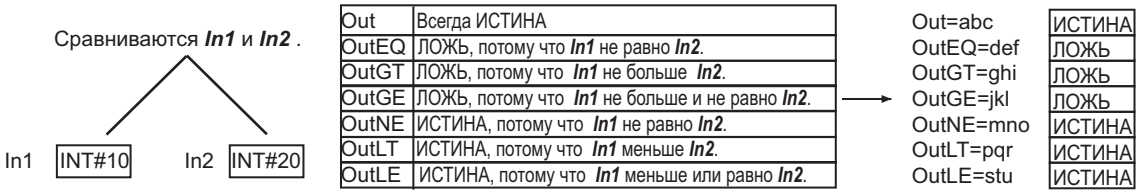
LD



ST

```
abc:=Cmp(INT#10, INT#20, def, ghi, jkl, mno, pqr, stu);
```

Команда Cmp сравнивает ***In1*** и ***In2***.
Ниже приведены результаты для разных условий сравнения.



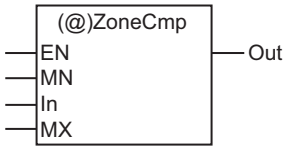
Меры предосторожности для обеспечения надлежащей эксплуатации

- Если типы данных значений *In1* и *In2* отличаются, перед сравнением один из типов данных будет приведен к другому типу данных, чтобы поддерживать все возможные значения обоих типов данных.
- Если *In1* и *In2* являются вещественными числами и содержат какое-либо десятичное число с бесконечной дробной частью, может произойти ошибка и результат сравнения может быть неправильным.
- Целые значения со знаком (SINT, INT, DINT и LINT) невозможно сравнивать с целыми значениями без знака (USINT, UINT, UDINT и ULINT).

- Два значения, являющихся положительной бесконечностью, равно как и два значения, являющихся отрицательной бесконечностью, считаются равными.
- Если значение *In1* или *In2* является нечисловым, выходы *OutEQ*, *OutGT*, *OutGE*, *OutNE*, *OutLT* и *OutLE* содержат ЛОЖЬ.

ZoneCmp

Команда ZoneCmp определяет, находится ли сравниваемое значение между указанными максимальным и минимальным значениями.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ZoneCmp	Попадание в диапазон	FUN		Out:=ZoneCmp(MN, In, MX);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
MN	Минимальное значение	Вход	Минимальное значение	Зависит от типа данных.	---	0
In	Сравниваемое значение		Значение для сравнения			*1
MX	Максимальное значение		Максимальное значение			0
Out	Результат сравнения	Выход	Результат сравнения	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MN						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	OK *1	OK *1	OK *1	OK *1	
In						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	OK *1	OK *1	OK *1	OK *1	
MX						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	OK *1	OK *1	OK *1	OK *1	
Out	ОК																			

*1. При использовании модуля ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше можно указать значение следующего типа: TIME, DATE, TOD и DT.

Функция

Команда ZoneCmp определяет, находится ли сравниваемое значение (*In*) между указанными максимальным (*MX*) и минимальным (*MN*) значениями.

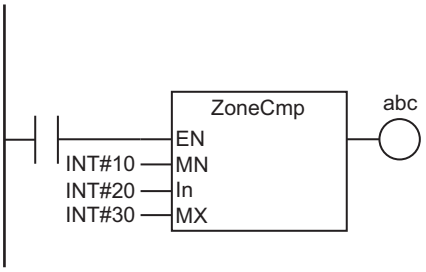
Если *MX* ≥ *In* ≥ *MN*, выход *Out* будет равен ИСТИНА. Иначе он будет содержать ЛОЖЬ.

Как соотносятся между собой значения, не являющиеся целыми или вещественными числами, поясняется в таблице ниже.

Тип данных	Соотношение
TIME	За большее принимается значение, которое больше в числовом выражении.
DATE, TOD или DT	За большее принимается более поздняя дата или время суток.

Ниже приведен пример, в котором *MN* = INT#10, *In* = INT#20, а *MX* = INT#30. Значение переменной *abc* будет равно ИСТИНА.

LD

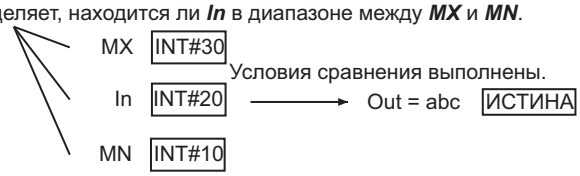


ST

```
abc:=ZoneCmp(INT#10, INT#20, INT#30);
```

Команда ZoneCmp определяет, выполняется ли условие: ***MX*** ≥ ***In*** ≥ ***MN***.
Если условия сравнения соблюдаются, значение ***abc*** будет равно ИСТИНА.

Команда определяет, находится ли ***In*** в диапазоне между ***MX*** и ***MN***.



Дополнительная информация

При сравнении значений типа TIME, DT или TOD необходимо определять их точность, чтобы сравнение выполнялось с такой же точностью. Для определения точности значений можно использовать следующие команды: *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 и *TruncTod* на стр. 2-757.

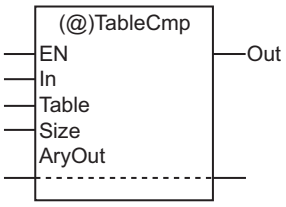
Меры предосторожности для обеспечения надлежащей эксплуатации

- Если типы данных значений *In*, *MX* и *MN* отличаются, перед сравнением они будут приведены к типу данных, который охватывает любые возможные значения каждого из используемых типов данных.
- Если *In*, *MX* и *MN* являются вещественными числами и содержат какое-либо десятичное число с бесконечной дробной частью, может произойти ошибка и результат сравнения может быть неправильным.
- Целые значения со знаком (SINT, INT, DINT и LINT) невозможно сравнивать с целыми значениями без знака (USINT, UINT, UDINT и ULINT).
- В случае типов данных TIME, DATE, TOD и DT допускается сравнивать только значения одного типа. Если типы данных указанных для сравнения переменных будут отличаться, произойдет ошибка сборки.

- Два значения, являющихся положительной бесконечностью, равно как и два значения, являющихся отрицательной бесконечностью, считаются равными.
- Если вход *In* содержит нечисловое значение, значение выхода *Out* равно ЛОЖЬ.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.
- В указанных ниже случаях происходит ошибка. *Out* переходит в состояние ЛОЖЬ.
  a) Значение *MN* больше значения *MX*.
  b) Переменная *MX* или *MN* содержит нечисловое значение.

TableCmp

Команда TableCmp проверяет, принадлежит ли заданное значение тому или иному диапазону, которые заданы в таблице сравнения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TableCmp	Таблица сравнения	FUN		Out:=TableCmp(In, Table, Size, AryOut);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Сравниваемое значение	Вход	Значение для сравнения	Зависит от типа данных.	---	*1
Table[] (двумерный массив)	Таблица сравнения		Двумерный массив, элементы которого используются для определения диапазонов.			
Size	Размер таблицы сравнения		Количество элементов в массиве Table[], с которыми требуется сравнить *In*			1
AryOut[] (массив)	Массив с отдельными результатами сравнения	Вход-выход	Результаты сравнения для элементов массива Table[] ИСТИНА: условие выполнено. ЛОЖЬ: условие не выполнено.	Зависит от типа данных.	---	---
Out	Результат сравнения	Выход	ИСТИНА: *In* удовлетворяет всем условиям сравнения для элементов массива Table[]. ЛОЖЬ: условие сравнения не выполняется для одного или нескольких наборов элементов.	Зависит от типа данных.	---	---

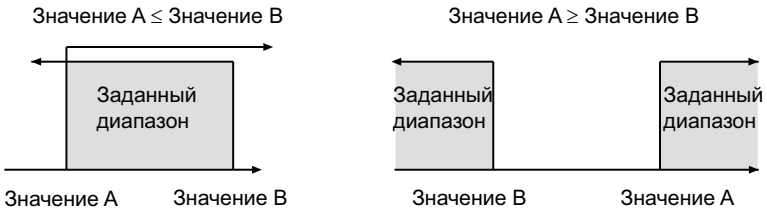
*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
Table[] (двумерный массив)	Должен представлять собой двумерный массив, элементы которого должны иметь тот же тип данных, что и значение *In*.																			
Size							ОК													
AryOut[] (массив)	ОК																			
Out	ОК																			

Функция

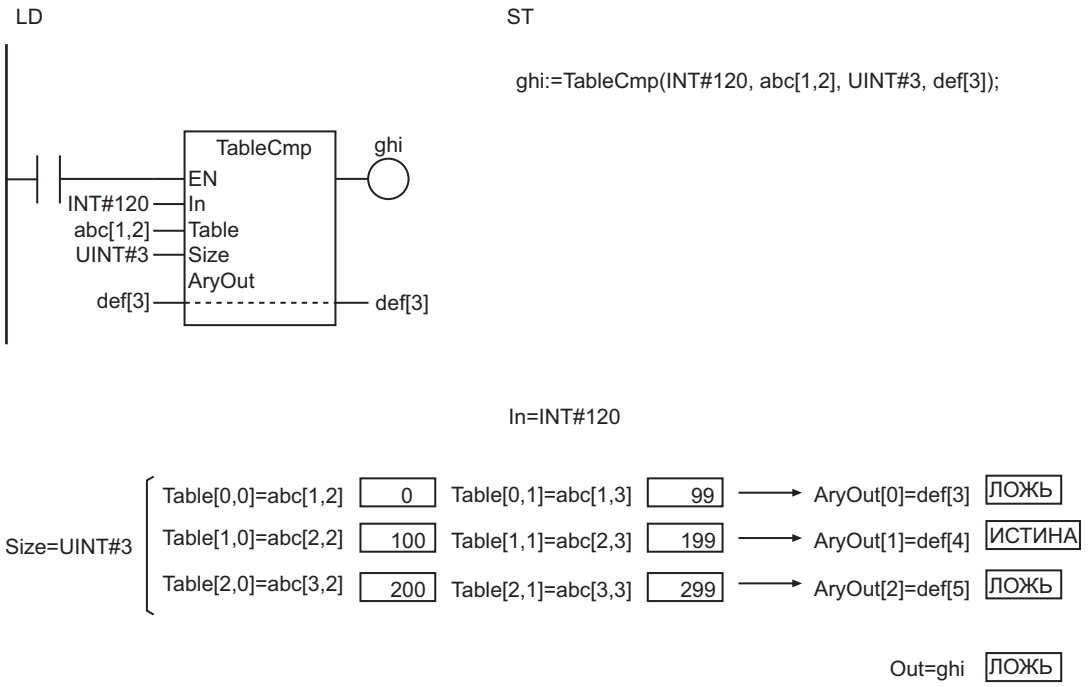
Команда TableCmp проверяет входное значение *In* на принадлежность диапазонам, заданным в таблице сравнения Table[]. Количество диапазонов задается параметром *Size*.
Параметр Table[] представляет собой двумерный массив. Первое измерение содержит номера заданных диапазонов. Во втором измерении элемент 0 задает значение A, а элемент 1 задает значение B соответствующего диапазона.

Заданные значения A и B определяют диапазон так, как показано на рисунке ниже. Значения A и B всегда включается в диапазон.



Результаты сравнения значения *In* с массивом Table[] записываются в массив индивидуальных результатов сравнения AryOut[]. Если значение *In* находится в пределах заданного диапазона для элемента i, элемент AryOut[i] содержит значение ИСТИНА. Если оно лежит вне диапазона, то AryOut[i] будет равно ЛОЖЬ. Если все (т. е. *Size*) элементы массива AryOut[] равны ИСТИНА, результат сравнения *Out* равен ИСТИНА. Иначе он будет содержать ЛОЖЬ.

Ниже приведен пример, в котором *In* = INT#120, а *Size* = UINT#3.



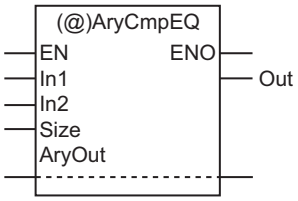
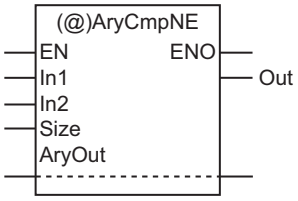
Меры предосторожности для обеспечения надлежащей эксплуатации

- Значение *In* и значения в Table[] должны принадлежать к одному типу данных. В противном случае произойдет ошибка сборки.
- Массив Table[] должен быть двумерным.
- Если для Table[] используется массив с числом измерений больше двух, элементы в третьем и более высоких измерениях игнорируются.
- Если длина массива AryOut[] превышает значение *Size*, результаты сравнения будут сохранены в элементы AryOut[0]...AryOut[*Size*-1]. Остальные элементы массива не изменятся.
- Целые значения со знаком (SINT, INT, DINT и LINT) невозможно сравнивать с целыми значениями без знака (USINT, UINT, UDINT и ULINT).
- При сравнении вещественных чисел может произойти ошибка и результат сравнения может быть неправильным. Это может произойти, например, если сравниваемые значения являются десятичными числами с бесконечной дробной частью.
- Если *Size* = 0, выход *Out* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.
- В указанных ниже случаях происходит ошибка. *Out* переходит в состояние ЛОЖЬ.
  a) Если значение *Size* превышает длину массива AryOut[].
  b) Если значение *Size* превышает длину первого измерения массива Table[].
  c) Если длина второго измерения массива Table[] равна 1.

AryCmpEQ и AryCmpNE

Эти команды сравнивают соответствующие элементы двух массивов.

AryCmpEQ : Определяет, равны ли друг другу соответствующие элементы двух массивов.
AryCmpNE : Определяет, отличаются ли друг от друга соответствующие элементы двух массивов.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryCmpEQ	Сравнение массивов: равно	FUN		AryCmpEQ(In1, In2, Size, AryOut);
AryCmpNE	Сравнение массивов: не равно	FUN		AryCmpNE(In1, In2, Size, AryOut);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1[] и In2[] (массивы)	Сравниваемые массивы	Вход	Массивы, содержащие элементы для сравнения	Зависит от типа данных.	---	*1
Size	Количество сравниваемых элементов		Количество элементов для сравнения			1
AryOut[] (массив)	Массив с результатами сравнения	Вход-выход	Массив с результатами сравнения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
In2[] (массив)	Должен быть массивом с тем же типом данных, что и In1[].																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size							ОК													
AryOut[] (массив)	ОК																			
Out	ОК																			

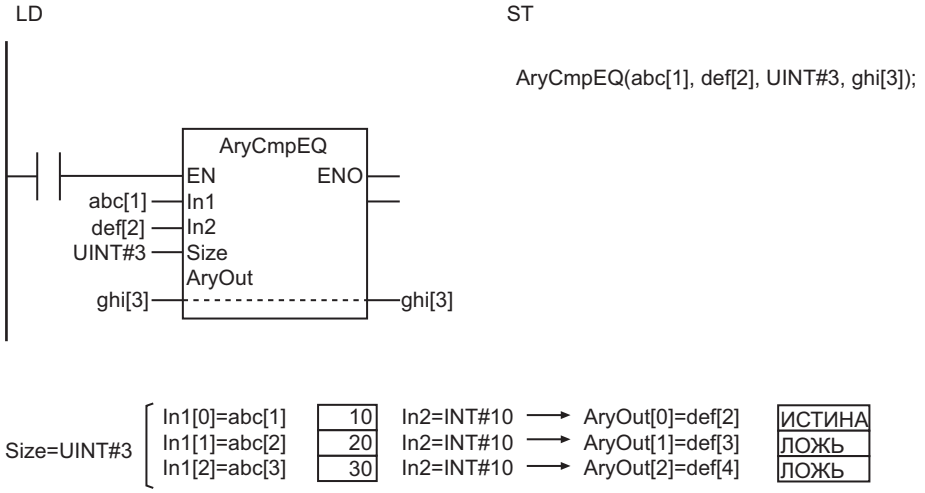
Функция

Эти команды служат для поэлементного сравнения двух массивов. То есть они сравнивают элементы с одинаковыми индексами: In1[0] с In2[0]...In1[*Size*-1] с In2[*Size*-1]. Результаты сравнения записываются в соответствующие элементы массива результатов сравнения AryOut[] (т. е. в элементы AryOut[0]...AryOut[*Size*-1]).

Ниже показано изменение значения AryOut[i] для каждой команды:

Команда	Значение AryOut[i]
AryCmpEQ	Если In1[i] = In2[i], то результат равен ИСТИНА. Иначе — ЛОЖЬ.
AryCmpNE	Если In1[i] ≠ In2[i], то результат равен ИСТИНА. Иначе — ЛОЖЬ.

Ниже показан пример работы команды AryCmpEQ для случая, когда *Size* = UINT#3.



Меры предосторожности для обеспечения надлежащей эксплуатации

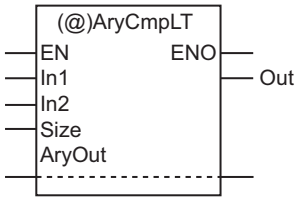
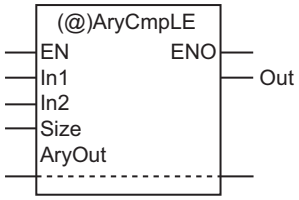
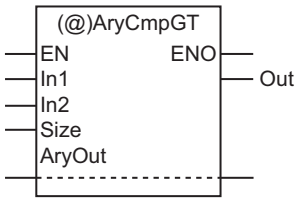
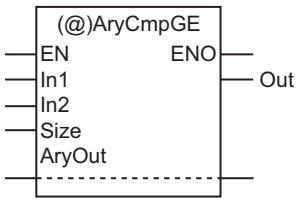
- Для In1[] и In2[] следует использовать один и тот же тип данных. При использовании разных типов данных произойдет ошибка сборки.
- Длина массива AryOut[] должна быть не меньше значения *Size*.
- Если массивы In1[] и In2[] содержат вещественные числа, при сравнении может произойти ошибка и результат сравнения может быть неправильным. Это может произойти, например, если сравниваемые значения являются десятичными числами с бесконечной дробной частью.

- Если *Size* = *0*, выход *Out* будет содержать ИСТИНА, а содержимое AryOut[] не изменится.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае происходит ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Если длина массива In1[], In2[] или AryOut[] меньше значения параметра *Size*.

AryCmpLT, AryCmpLE, AryCmpGT и AryCmpGE

Эти команды сравнивают соответствующие элементы двух массивов указанным ниже образом.

AryCmpLT : Выполняет сравнение «меньше».
AryCmpLE : Выполняет сравнение «меньше или равно».
AryCmpGT : Выполняет сравнение «больше».
AryCmpGE : Выполняет сравнение «больше или равно».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryCmpLT	Сравнение массивов: меньше	FUN		AryCmpLT(In1, In2, Size, AryOut);
AryCmpLE	Сравнение массивов: меньше или равно	FUN		AryCmpLE(In1, In2, Size, AryOut);
AryCmpGT	Сравнение массивов: больше	FUN		AryCmpGT(In1, In2, Size, AryOut);
AryCmpGE	Сравнение массивов: больше или равно	FUN		AryCmpGE(In1, In2, Size, AryOut);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1[] и In2[] (массивы)	Сравниваемые массивы	Вход	Массивы, содержащие элементы для сравнения	Зависит от типа данных.	---	*1
Size	Количество сравниваемых элементов		Количество элементов для сравнения			1

	Назначение	Вход/выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
AryOut[] (массив)	Массив с результатами сравнения	Вход-выход	Массив с результатами сравнения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
In2[] (массив)	Должен быть массивом с тем же типом данных, что и In1[].																			
Size							ОК													
AryOut[] (массив)	ОК																			
Out	ОК																			

Функция

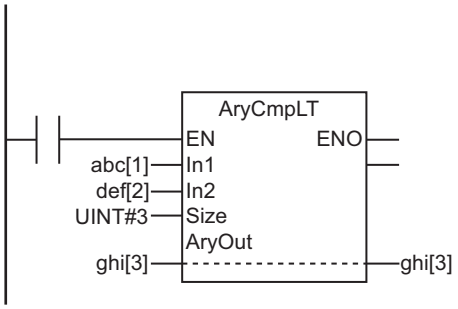
Эти команды служат для поэлементного сравнения двух массивов. То есть они сравнивают элементы с одинаковыми индексами: In1[0] с In2[0]...In1[*Size*-1] с In2[*Size*-1]. Результаты сравнения записываются в соответствующие элементы массива результатов сравнения AryOut[] (т. е. в элементы AryOut[0]...AryOut[*Size*-1]).

Ниже показано изменение значения AryOut[i] для каждой команды:

Команда	Значение AryOut[i]
AryCmpLT	Если In1[i] < In2[i], то результат равен ИСТИНА. Иначе — ЛОЖЬ.
AryCmpLE	Если In1[i] <= In2[i], то результат равен ИСТИНА. Иначе — ЛОЖЬ.
AryCmpGT	Если In1[i] > In2[i], то результат равен ИСТИНА. Иначе — ЛОЖЬ.
AryCmpGE	Если In1[i] >= In2[i], то результат равен ИСТИНА. Иначе — ЛОЖЬ.

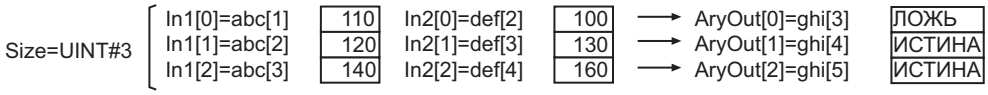
Ниже показан пример работы команды AryCmpLT для случая, когда *Size* = UINT#3.

LD



ST

```
AryCmpLT(abc[1], def[2], UINT#3, ghi[3]);
```



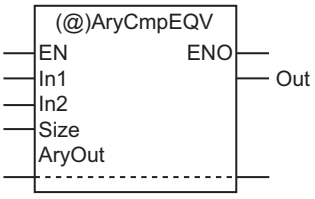
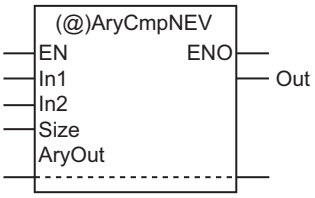
Меры предосторожности для обеспечения надлежащей эксплуатации

- Для In1[] и In2[] следует использовать один и тот же тип данных. При использовании разных типов данных произойдет ошибка сборки.
- Длина массива AryOut[] должна быть не меньше значения *Size*.
- Если массивы In1[] и In2[] содержат вещественные числа, при сравнении может произойти ошибка и результат сравнения может быть неправильным. Это может произойти, например, если сравниваемые значения являются десятичными числами с бесконечной дробной частью.
- Если *Size = 0*, выход *Out* будет содержать ИСТИНА, а содержимое AryOut[] не изменится.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае происходит ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Если длина массива In1[], In2[] или AryOut[] меньше значения параметра *Size*.

AryCmpEQV и AryCmpNEV

Эти команды сравнивают каждый элемент массива с заданным значением.

AryCmpEQV : Определяет, равен ли каждый элемент массива заданному значению.
AryCmpNEV : Определяет, отличается ли каждый элемент массива от заданного значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryCmpEQV	Сравнение массива со значением: равно	FUN		AryCmpEQV(In1, In2, Size, AryOut);
AryCmpNEV	Сравнение массива со значением: не равно	FUN		AryCmpNEV(In1, In2, Size, AryOut);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1[] (массив)	Сравниваемый массив	Вход	Массив, содержащий элементы для сравнения	Зависит от типа данных.	---	*1
In2	Сравниваемое значение		Значение для сравнения			
Size	Количество сравниваемых элементов		Количество элементов для сравнения			1
AryOut[] (массив)	Массив с результатами сравнения	Вход-выход	Массив с результатами сравнения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In2	Тип данных должен быть таким же, как у элементов массива In1[].																			
Size							OK													
AryOut[] (массив)	OK																			
Out	OK																			

Функция

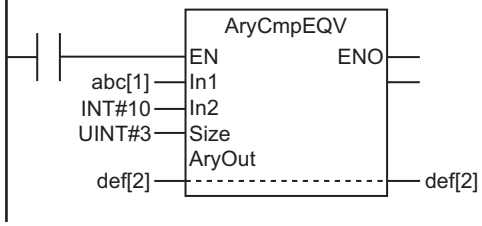
Эти команды сравнивают заданное значение *In2* с указанными элементами массива (In1[0]...In1[*Size*-1]).
Результаты сравнения записываются в соответствующие элементы массива результатов сравнения AryOut[] (т. е. в элементы AryOut[0]...AryOut[*Size*-1]).

Ниже показано изменение значения AryOut[i] для каждой команды:

Команда	Значение AryOut[i]
AryCmpEQV	Если In1[i] = In2, то результат равен ИСТИНА. Иначе — ЛОЖЬ.
AryCmpNEV	Если In1[i] ≠ In2, то результат равен ИСТИНА. Иначе — ЛОЖЬ.

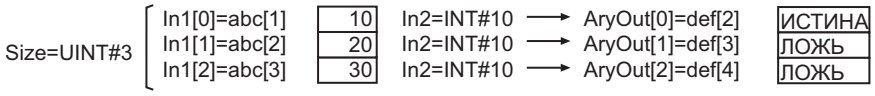
Ниже приведен пример использования команды AryCmpEQV для случая, когда *In2* = INT#10, а *Size* = UINT#3.

LD



ST

```
AryCmpEQV(abc[1], INT#10, UINT#3, def[2]);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

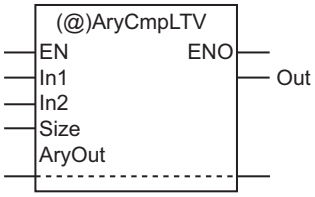
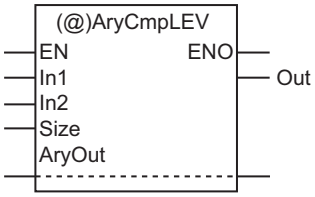
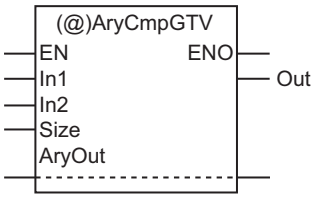
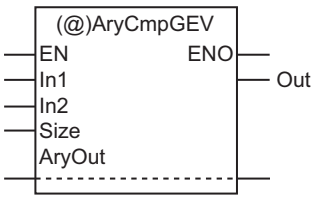
- Для In1[] и *In2* следует использовать один и тот же тип данных. При использовании разных типов данных произойдет ошибка сборки.
- Длина массива AryOut[] должна быть не меньше значения *Size*.

- Если массив In1[] содержит вещественные числа и *In2* также является вещественным числом, при сравнении может произойти ошибка и результат сравнения может быть неправильным. Это может произойти, например, если сравниваемые значения являются десятичными числами с бесконечной дробной частью.
- Если *Size = 0*, выход *Out* будет содержать ИСТИНА, а содержимое AryOut[] не изменится.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае происходит ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Если длина массива In1[] или AryOut[] меньше значения параметра *Size*.

AryCmpLTV, AryCmpLEV, AryCmpGTV и AryCmpGEV

Эти команды сравнивают каждый элемент массива с заданным значением указанным ниже образом.

AryCmpLTV : Выполняет сравнение «меньше».
AryCmpLEV : Выполняет сравнение «меньше или равно».
AryCmpGTV : Выполняет сравнение «больше».
AryCmpGEV : Выполняет сравнение «больше или равно».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryCmpLTV	Сравнение массива со значением: меньше	FUN		AryCmpLTV(In1, In2, Size, AryOut);
AryCmpLEV	Сравнение массива со значением: меньше или равно	FUN		AryCmpLEV(In1, In2, Size, AryOut);
AryCmpGTV	Сравнение массива со значением: больше	FUN		AryCmpGTV(In1, In2, Size, AryOut);
AryCmpGEV	Сравнение массива со значением: больше или равно	FUN		AryCmpGEV(In1, In2, Size, AryOut);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1[] (массив)	Сравниваемый массив	Вход	Массив, содержащий элементы для сравнения	Зависит от типа данных.	---	*1
In2	Сравниваемое значение		Значение для сравнения			
Size	Количество сравниваемых элементов		Количество элементов для сравнения			1
AryOut[] (массив)	Массив с результатами сравнения	Вход-выход	Массив с результатами сравнения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
In2	Тип данных должен быть таким же, как у элементов массива In1[].																			
Size							ОК													
AryOut[] (массив)	ОК																			
Out	ОК																			

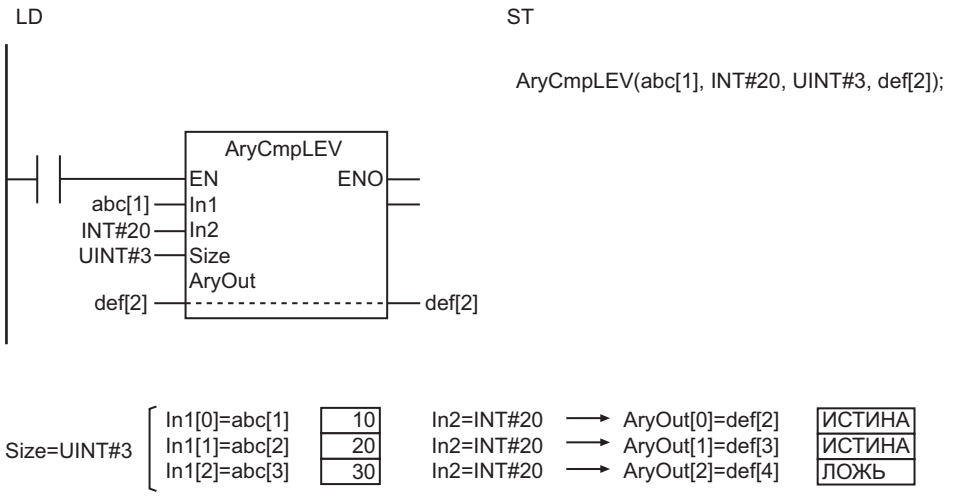
Функция

Эти команды сравнивают заданное значение *In2* с указанными элементами массива (In1[0]...In1[*Size*-1]).
Результаты сравнения записываются в соответствующие элементы массива результатов сравнения AryOut[] (т. е. в элементы AryOut[0]...AryOut[*Size*-1]).

Ниже показано изменение значения AryOut[i] для каждой команды:

Команда	Значение AryOut[i]
AryCmpLTV	Если In1[i] < In2, то результат равен ИСТИНА. Иначе — ЛОЖЬ.
AryCmpLEV	Если In1[i] <= In2, то результат равен ИСТИНА. Иначе — ЛОЖЬ.
AryCmpGTV	Если In1[i] > In2, то результат равен ИСТИНА. Иначе — ЛОЖЬ.
AryCmpGEV	Если In1[i] >= In2, то результат равен ИСТИНА. Иначе — ЛОЖЬ.

Ниже приведен пример использования команды AryCmpLEV для случая, когда *In2* = INT#20, а *Size* = UINT#3.



Меры предосторожности для обеспечения надлежащей эксплуатации

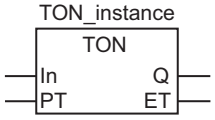
- Для In1[] и *In2* следует использовать один и тот же тип данных. При использовании разных типов данных произойдет ошибка сборки.
- Длина массива AryOut[] должна быть не меньше значения *Size*.
- Если массив In1[] содержит вещественные числа и *In2* также является вещественным числом, при сравнении может произойти ошибка и результат сравнения может быть неправильным. Это может произойти, например, если сравниваемые значения являются десятичными числами с бесконечной дробной частью.
- Если *Size = 0*, выход *Out* будет содержать ИСТИНА, а содержимое AryOut[] не изменится.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае происходит ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Если длина массива In1[] или AryOut[] меньше значения параметра *Size*.

Команды таймеров

Команда	Имя	Стр.
TON	Таймер задержки включения	стр. 2-148
TOF	Таймер задержки выключения	стр. 2-154
TP	Таймер импульса	стр. 2-158
AccumulationTimer	Накопительный таймер	стр. 2-162
Timer	100 мс таймер	стр. 2-166

TON

Команда TON выдает состояние «ИСТИНА» по истечении заданного времени с момента запуска таймера.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TON	Таймер задержки включения	FB		TON_instance (In, PT, Q,ET);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Вход таймера	Вход	ИСТИНА: сигнал запуска таймера ЛОЖЬ: сигнал сброса таймера	Зависит от типа данных.	---	ЛОЖЬ
PT	Установленное время		Время с момента запуска таймера, по истечении которого выход *Q* переходит в состояние «ИСТИНА».	*1	мс	0
Q	Выход таймера	Выход	ИСТИНА: выход таймера включен ЛОЖЬ: выход таймера выключен	Зависит от типа данных.	---	---
ET	Истекшее время		Время, прошедшее с момента запуска таймера	*1	мс	

*1. T#0 мс...T#106751д_23ч_47м_16с_854,775807мс

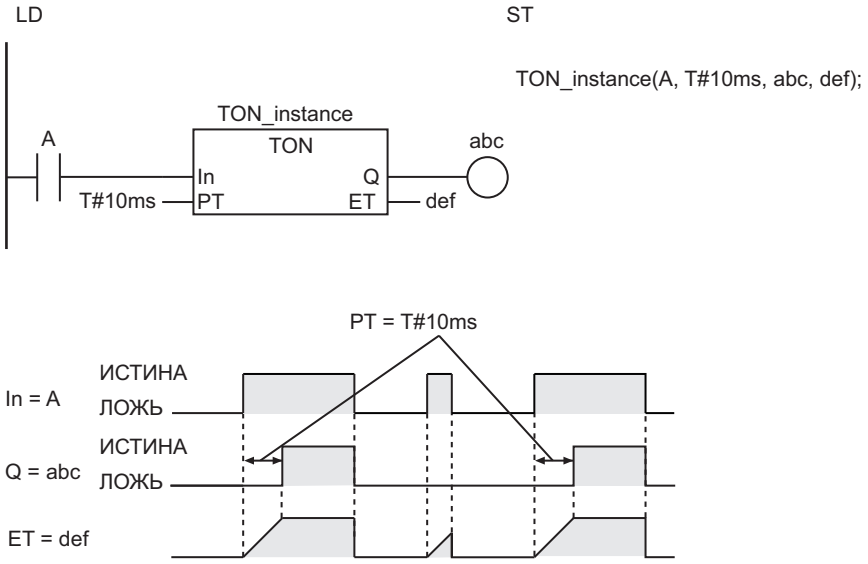
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	ОК																			
PT																ОК				
Q	ОК																			
ET																ОК				

Функция

Команда TON выдает состояние «ИСТИНА» по истечении заданного времени с момента запуска таймера. Время задается в наносекундах.
Таймер начинает работать, когда вход таймера *In* переходит в состояние ИСТИНА. По мере отсчета времени значение переменной *ET* (истекшее время) увеличивается.
Когда значение *ET* достигает заданного времени *PT*, выход таймера *Q* переходит в состояние ИСТИНА. После этого *ET* больше не увеличивается.
Когда вход *In* переходит в состояние ЛОЖЬ, таймер сбрасывается. *ET* принимает значение 0, а на выходе *Q* устанавливается состояние ЛОЖЬ.

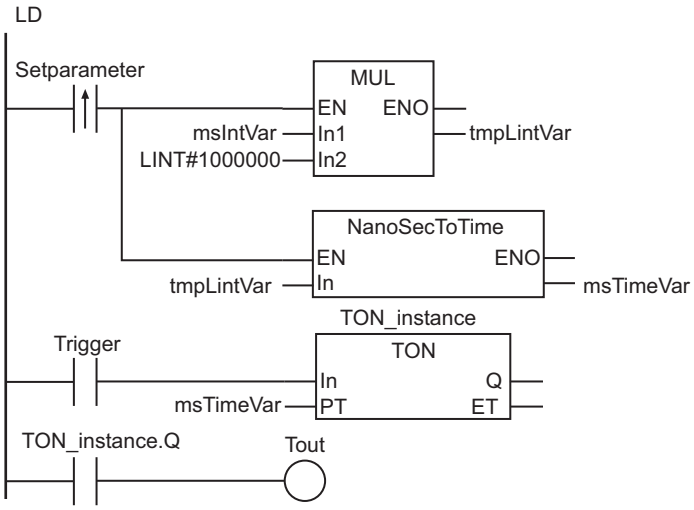
Если вход *In* переходит в состояние ЛОЖЬ после запуска таймера, таймер сбрасывается, даже если переменная *ET* еще не достигла значения *PT*.

На рисунке ниже показаны пример программы и временная диаграмма для случая, когда *PT* = T#10 мс. Переменная *abc* примет значение ИСТИНА через 10 мс после того, как значение переменной *A* поменяется на ИСТИНА.



Дополнительная информация

- Если в программе нужно применить таймер, выход которого переходит в состояние ИСТИНА, когда начинается отсчет времени, и возвращается в состояние ЛОЖЬ по достижении заданного времени, используйте команду *TP* на стр. 2-158.
- Если требуется таймер, который начинает отсчет времени, когда вход *In* переходит в состояние ЛОЖЬ, и устанавливает на своем выходе состояние ЛОЖЬ по истечении заданного времени, используйте команду *TOF* на стр. 2-154.
- Если требуется, чтобы время выполнения команды таймера было меньше, используйте команду *Timer* на стр. 2-166, которая измеряет время с шагом в 100 мс.
- В случае подключения к устройству HMI, которое не поддерживает тип данных TIME, перед вводом заданного целочисленного значения времени в эту команду его необходимо преобразовать в тип данных TIME. Для преобразования целочисленного значения в значение типа TIME используйте команду *NanoSecToTime* на стр. 2-727. Для преобразования значения типа TIME в целочисленное значение используйте команду *TimeToNanoSec* на стр. 2-723. В обеих командах время выражается в наносекундах. В представленном ниже примере программы переменная *msIntVar* типа INT — это установленное время в миллисекундах.



ST

```
tmpLintVar:=msIntVar*LINT#1000000;
msTimeVar:=NanoSecToTime(tmpLintVar);
TON_instance(In:=Trigger, PT:=msTimeVar, Q=>Tout);
```

Меры предосторожности для обеспечения надлежащей эксплуатации

- Ошибка отсчета времени, при которой выход *Q* = ИСТИНА для *PT*: от -100 нс до (100 нс + 1 период задачи).
  Вышеуказанный диапазон включает следующее:
  a) ±100 нс — ошибка отчета истекшего времени *ET*.
  b) В каждом периоде выполнения задачи проверяется, достигло ли время *ET* значения *PT*. Если время *ET* достигает значения *PT* сразу после выполнения данной проверки, это приводит к задержке длительностью в один период задачи.
- В Sysmac Studio время отображается с шагом 0,001 мс, но точность отсчета времени составляет 1 нс.
- Если на входе *In* уже присутствует состояние ИСТИНА, таймер запускается, как только начинается работа.
- Если для параметра *PT* задано значение T#0 мс или отрицательное число, выход *Q* перейдет в состояние ИСТИНА сразу, как только значение *In* поменяется на ИСТИНА.
- Значение *PT* можно изменять, когда вход *In* находится в состоянии ИСТИНА. Команда работает следующим образом:

Состояние таймера	Значение *Q*	Значение *PT* после изменения	Работа
После завершения отсчета времени	ИСТИНА	---	*Q* остается в состоянии ИСТИНА. Значение *ET* также не изменяется. Оно остается равным значению *PT* до изменения.
Выполняется отсчет времени	ЛОЖЬ	PT ≥ ET	Отсчет времени продолжается. Когда значение *ET* достигает значения *PT*, значение *Q* меняется на ИСТИНА и *ET* больше не увеличивается.
		PT < ET	Значение *Q* немедленно меняется на ИСТИНА. Увеличение *ET* немедленно прекращается.

- Если эта команда находится в области главного управления и область главного управления сброшена, таймер сбрасывается. Значение *ET* становится равно 0, а значение *Q* меняется на ЛОЖЬ.
- Если эта команда не выполняется из-за выполнения какой-либо команды перехода (например, команды JMP), значение *ET* не обновляется. Однако отсчет времени по-прежнему продолжается. Поэтому при следующем выполнении данной команды переменная *ET* принимает правильное значение.
- При использовании этой команды в лестничной диаграмме значение выхода *Q* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

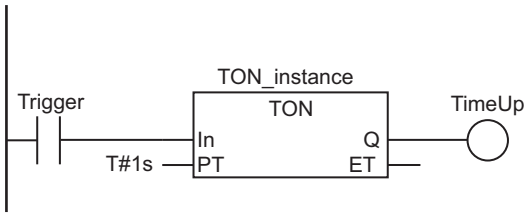
Пример программы

Измерение времени с помощью одного таймера задержки включения

Значение TimeUp меняется на ИСТИНА через одну секунду после перехода Trigger в состояние ИСТИНА.

● Программа на языке LD

Переменная	Тип данных	Начальное значение	Комментарий
Trigger	BOOL	ЛОЖЬ	Условие выполнения
TimeUp	BOOL	ЛОЖЬ	Выход таймера
TON_instance	TON		



● Программа на языке ST (пример 1)

Переменная	Тип данных	Начальное значение	Комментарий
Trigger	BOOL	ЛОЖЬ	Условие выполнения
TimeUp	BOOL	ЛОЖЬ	Выход таймера
TON_instance	TON		

```
IF (Trigger=TRUE) THEN
    TON_instance(In:=TRUE, PT:=T#1s, Q=>TimeUp);
ELSE
    TON_instance(In:=FALSE, Q=>TimeUp);
END_IF;
```

● Программа на языке ST (пример 2)

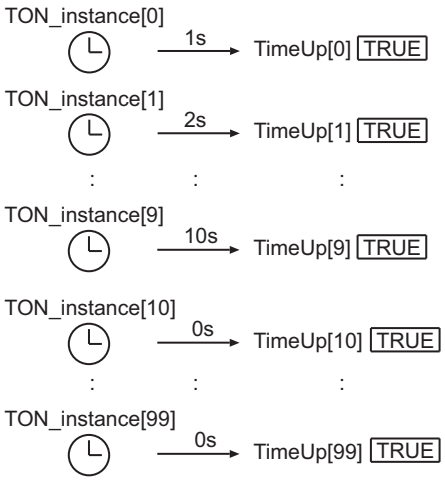
Переменная	Тип данных	Начальное значение	Комментарий
Trigger	BOOL	ЛОЖЬ	Условие выполнения
TimeUp	BOOL	ЛОЖЬ	Выход таймера

Переменная	Тип данных	Начальное значение	Комментарий
TON_instance	TON		

```
TON_instance(In:=Trigger, PT:=T#1s, Q=>TimeUp);
```

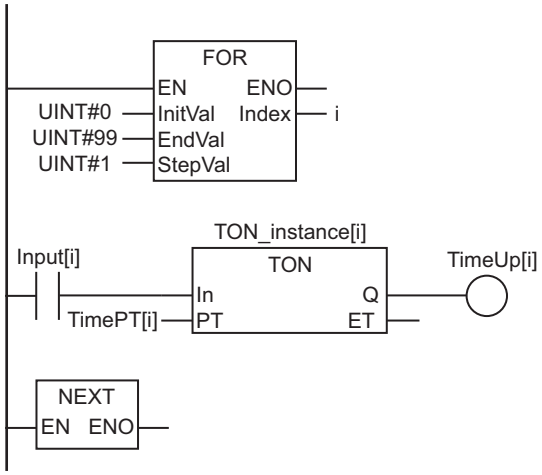
Измерение времени с помощью нескольких таймеров задержки включения

В данном примере в программе используется 100 экземпляров команды On-Delay Timer: TON_instance[0]...TON_instance[99]. Каждый таймер запускается, когда значение соответствующего входа таймера (Input[0]...Input[99]) меняется на ИСТИНА.
Первые 10 экземпляров таймеров (TON_instance[0]...TON_instance[9]) переводят соответствующие выходы TimeUp[i] (i = 0...9) в состояние ИСТИНА через i+1 секунд после начала выполнения.
Остальные 90 экземпляров таймеров (TON_instance[10]...TON_instance[99]) переводят соответствующие выходы TimeUp[i] (i = 10...99) в состояние ИСТИНА сразу после начала выполнения.



● Программа на языке LD

Переменная	Тип данных	Начальное значение	Комментарий
Input	ARRAY[0..99] OF BOOL	[100(ЛОЖЬ)]	Вход таймера
TimeUp	ARRAY[0..99] OF BOOL	[100(ЛОЖЬ)]	Выход таймера
TimePT	ARRAY[0..99] OF TIME	[T#1s, T#2s, T#3s, T#4s, T#5s, T#6s, T#7s, T#8s, T#9s, T#10s, 90(T#0s)]	Установленное время
TON_instance	ARRAY[0..99] OF TON		
i	UINT	0	Указатель



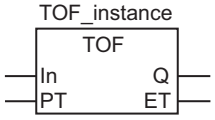
● Программа на языке ST

Переменная	Тип данных	Начальное значение	Коммента-рий
Input	ARRAY[0..99] OF BOOL	[100(ЛОЖЬ)]	Вход тайме-ра
TimeUp	ARRAY[0..99] OF BOOL	[100(ЛОЖЬ)]	Выход тай-мера
TimePT	ARRAY[0..99] OF TIME	[T#1s, T#2s, T#3s, T#4s, T#5s, T#6s, T#7s, T#8s, T#9s, T#10s, 90(T#0s)]	Установлен-ное время
TON_instance	ARRAY[0..99] OF TON		
i	UINT	0	Указатель

```
FOR i:=UINT#0 TO UINT#99 DO
    TON_instance[i](
        In := Input[i],
        PT := TimePT[i],
        Q =>TimeUp[i]);
END_FOR;
```

TOF

Команда TOF выдает состояние «ЛОЖЬ» по истечении заданного времени с момента запуска таймера.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TOF	Таймер задержки выключения	FB		TOF_instance(In, PT, Q, ET);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Вход таймера	Вход	ИСТИНА: сигнал сброса таймера ЛОЖЬ: сигнал запуска таймера	Зависит от типа данных.	---	ЛОЖЬ
PT	Установленное время		Время с момента запуска таймера, по истечении которого выход *Q* переходит в состояние «ИСТИНА».	*1	мс	0
Q	Выход таймера	Выход	ИСТИНА: выход таймера включен ЛОЖЬ: выход таймера выключен	Зависит от типа данных.	---	---
ET	Истекшее время		Время, прошедшее с момента запуска таймера	*1	мс	

*1. T#0 мс...T#106751д_23ч_47м_16с_854,775807мс

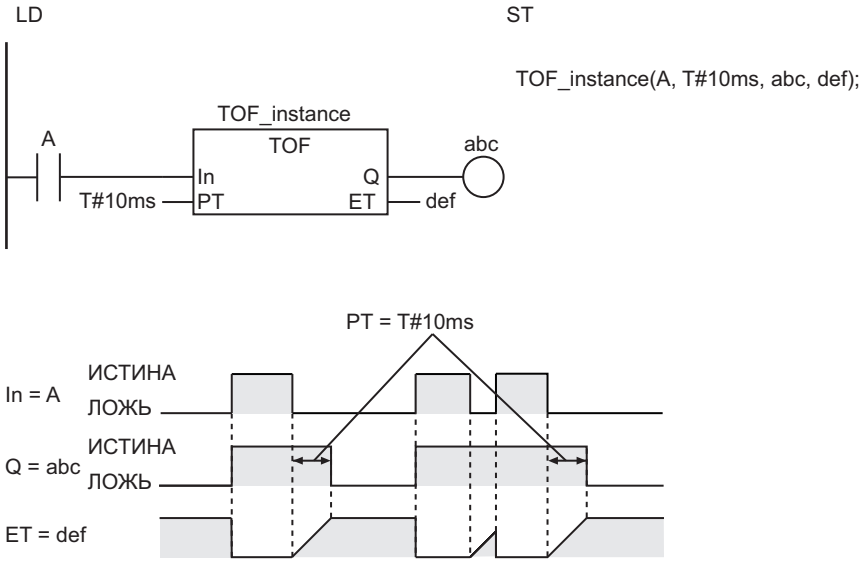
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	ОК																			
PT																ОК				
Q	ОК																			
ET																ОК				

Функция

Команда TOF выдает состояние «ЛОЖЬ» по истечении заданного времени с момента запуска таймера. Время задается в наносекундах.
Таймер начинает работать, когда вход таймера *In* переходит в состояние ЛОЖЬ. По мере отсчета времени значение переменной *ET* (истекшее время) увеличивается.
Когда значение *ET* достигает заданного времени *PT*, выход таймера *Q* переходит в состояние ЛОЖЬ. После этого *ET* больше не увеличивается.
Когда вход *In* устанавливается в состояние ИСТИНА, таймер сбрасывается. *ET* принимает значение 0, а на выходе *Q* устанавливается состояние ИСТИНА.

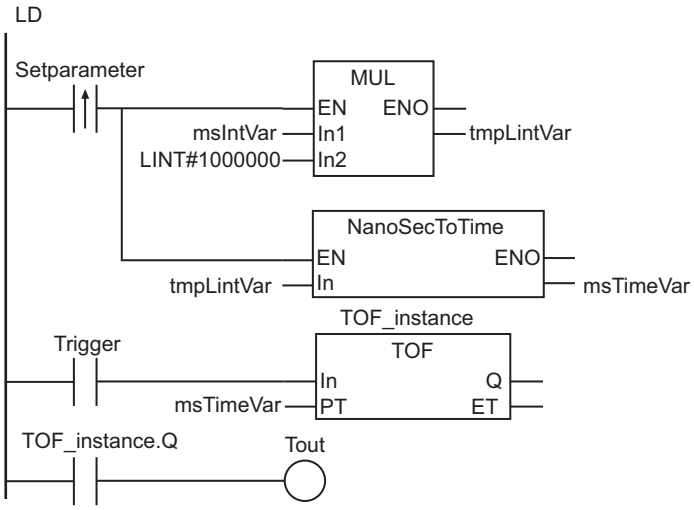
Если вход *In* переходит в состояние ИСТИНА после запуска таймера, таймер сбрасывается, даже если переменная *ET* еще не достигла значения *PT*.

На рисунке ниже показаны пример программы и временная диаграмма для случая, когда *PT* =T#10 мс. Переменная *abc* примет значение ЛОЖЬ через 10 мс после того, как значение переменной *A* поменяется на ЛОЖЬ.



Дополнительная информация

- Если в программе нужно применить таймер, выход которого переходит в состояние ИСТИНА, когда начинается отсчет времени, и возвращается в состояние ЛОЖЬ по достижении заданного времени, используйте команду *TP* на стр. 2-158.
- Если требуется таймер, который начинает отсчет времени, когда вход *In* переходит в состояние ИСТИНА, и устанавливает на своем выходе состояние ИСТИНА по истечении заданного времени, используйте команду *TON* на стр. 2-148.
- В случае подключения к устройству HMI, которое не поддерживает тип данных TIME, перед вводом заданного целочисленного значения времени в эту команду его необходимо преобразовать в тип данных TIME. Для преобразования целочисленного значения в значение типа TIME используйте команду *NanoSecToTime* на стр. 2-727. Для преобразования значения типа TIME в целочисленное значение используйте команду *TimeToNanoSec* на стр. 2-723. В обеих командах время выражается в наносекундах. В представленном ниже примере программы переменная msIntVar типа INT — это установленное время в миллисекундах.



ST

```
tmpLintVar:=msIntVar*LINT#1000000;
msTimeVar:=NanoSecToTime(tmpLintVar);
TOF_instance(In:=Trigger, PT:=msTimeVar, Q=>Tout);
```

Меры предосторожности для обеспечения надлежащей эксплуатации

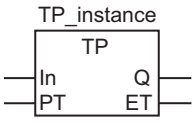
- Ошибка отсчета времени, при которой выход *Q* = ИСТИНА для *PT*: от -100 нс до (100 нс + 1 период задачи).
  Вышеуказанный диапазон включает следующее:
  a) ±100 нс — ошибка отчета истекшего времени *ET*.
  b) В каждом периоде выполнения задачи проверяется, достигло ли время *ET* значения *PT*. Если время *ET* достигает значения *PT* сразу после выполнения данной проверки, это приводит к задержке длительностью в один период задачи.
- В Sysmac Studio время отображается с шагом 0,001 мс, но точность отсчета времени составляет 1 нс.
- Если для параметра *PT* задано значение T#0 мс или отрицательное число, выход *Q* перейдет в состояние ЛОЖЬ сразу, как только значение *In* поменяется на ЛОЖЬ.
- После выполнения этой команды значение *Q* меняется на ИСТИНА, если значение *In* равно ИСТИНА. По истечении установленного времени *PT* после запуска таймера значение *Q* меняется на ЛОЖЬ.
- Значение *PT* можно изменять, когда вход *In* находится в состоянии ЛОЖЬ. Команда работает следующим образом:

Состояние таймера	Значение *Q*	Значение *PT* после изменения	Работа
После завершения отсчета времени	ЛОЖЬ	---	*Q* остается в состоянии ЛОЖЬ. Значение *ET* также не изменяется. Оно остается равным значению *PT* до изменения.
Выполняется отсчет времени	ИСТИНА	PT ≥ ET	Отсчет времени продолжается. Когда значение *ET* достигает значения *PT*, значение *Q* меняется на ЛОЖЬ и *ET* больше не увеличивается.
		PT < ET	Значение *Q* немедленно меняется на ЛОЖЬ. Увеличение *ET* немедленно прекращается.

- Если эта команда находится в области главного управления и область главного управления сброшена, команда работает следующим образом:
  a) Значение *ET* становится равно 0, а значение *Q* меняется на ИСТИНА.
  b) Если к выходу *Q* подключена команда Out, условие выполнения команды Out остается равным ЛОЖЬ.
  c) Отсчет времени начинается сразу после отмены сброса.
- Если эта команда не выполняется из-за выполнения какой-либо команды перехода (например, команды JMP), значение *ET* не обновляется. Однако отсчет времени по-прежнему продолжается. Поэтому при следующем выполнении данной команды переменная *ET* принимает правильное значение.
- При использовании этой команды в лестничной диаграмме значение выхода *Q* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

TP

Команда TP выдает состояние «ИСТИНА» в течение заданного времени с момента запуска таймера.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TP	Таймер импульса	FB		TP_instance(In, PT, Q, ET);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Вход таймера	Вход	ИСТИНА: сигнал запуска таймера ЛОЖЬ: сигнал сброса таймера	Зависит от типа данных.	---	ЛОЖЬ
PT	Установленное время		Время, в течение которого выход *Q* остается в состоянии ИСТИНА.	*1	мс	0
Q	Выход таймера	Выход	ИСТИНА: выход таймера включен ЛОЖЬ: выход таймера выключен	Зависит от типа данных.	---	---
ET	Истекшее время		Время, прошедшее с момента запуска таймера	*1	мс	

*1. T#0 мс...T#106751д_23ч_47м_16с_854,775807мс

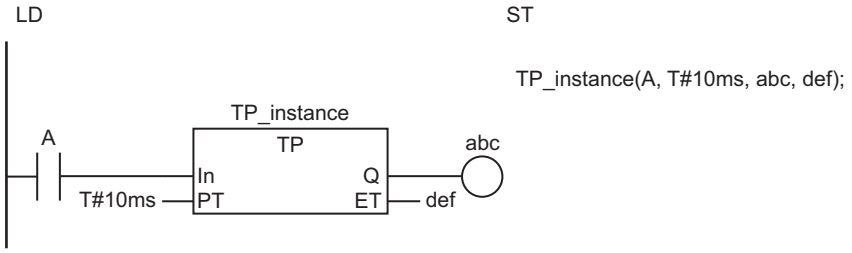
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	ОК																			
PT																ОК				
Q	ОК																			
ET																ОК				

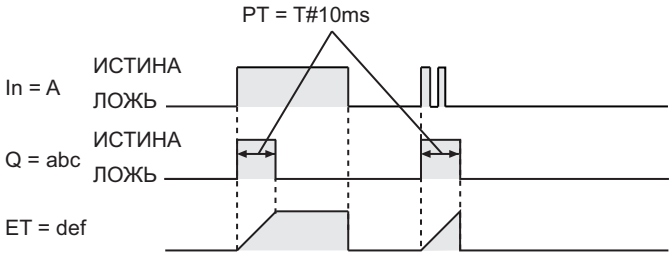
Функция

Команда TP выдает состояние «ИСТИНА» в течение заданного времени с момента запуска таймера. Время задается в наносекундах.

Когда вход таймера *In* переходит в состояние ИСТИНА, таймер начинает работать и выход таймера *Q* переходит в состояние ИСТИНА. По мере отсчета времени значение переменной *ET* (истекшее время) увеличивается.
Когда значение *ET* достигает заданного времени *PT*, выход таймера *Q* переходит в состояние ЛОЖЬ. После этого *ET* больше не увеличивается.
Когда вход *In* переходит в состояние ЛОЖЬ, таймер сбрасывается. *ET* сбрасывается в 0.
Если вход *In* переходит в состояние ЛОЖЬ прежде, чем *ET* достигает значения *PT*, таймер не сбрасывается.

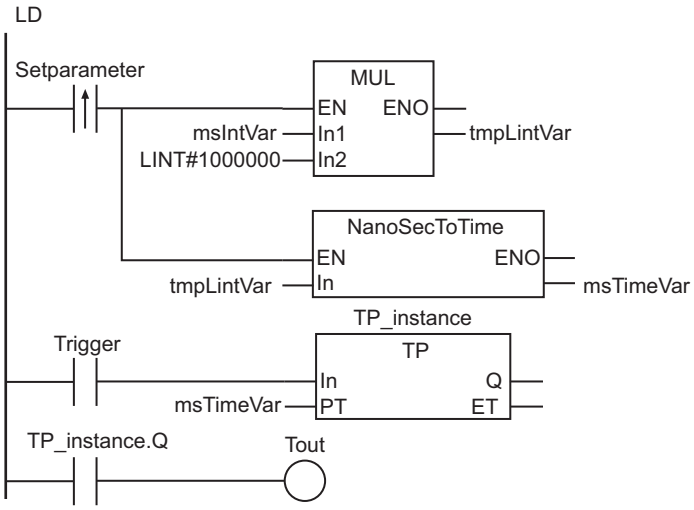
На рисунке ниже показаны пример программы и временная диаграмма для случая, когда *PT* = T#10 мс. После перехода переменной *A* в состояние ИСТИНА значение переменной *abc* меняется на ИСТИНА, а 10 мс спустя снова становится равно ЛОЖЬ.





Дополнительная информация

- Если требуется таймер, который начинает отсчет времени, когда вход *In* переходит в состояние ИСТИНА, и устанавливает на своем выходе состояние ИСТИНА по истечении заданного времени, используйте команду *TON* на стр. 2-148.
- Если требуется таймер, который начинает отсчет времени, когда вход *In* переходит в состояние ЛОЖЬ, и устанавливает на своем выходе состояние ЛОЖЬ по истечении заданного времени, используйте команду *TOF* на стр. 2-154.
- В случае подключения к устройству HMI, которое не поддерживает тип данных TIME, перед вводом заданного целочисленного значения времени в эту команду его необходимо преобразовать в тип данных TIME. Для преобразования целочисленного значения в значение типа TIME используйте команду *NanoSecToTime* на стр. 2-727. Для преобразования значения типа TIME в целочисленное значение используйте команду *TimeToNanoSec* на стр. 2-723. В обеих командах время выражается в наносекундах. В представленном ниже примере программы переменная *msIntVar* типа INT — это установленное время в миллисекундах.



ST

```
tmpLintVar:=msIntVar*LINT#1000000;
msTimeVar:=NanoSecToTime(tmpLintVar);
TP_instance(In:=Trigger, PT:=msTimeVar, Q=>Tout);
```

Меры предосторожности для обеспечения надлежащей эксплуатации

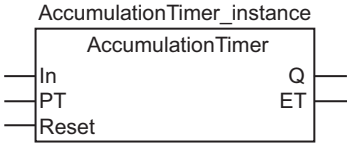
- Ошибка отсчета времени, при которой выход *Q* = ИСТИНА для *PT*: от -100 нс до (100 нс + 1 период задачи).
  Вышеуказанный диапазон включает следующее:
  a) ±100 нс — ошибка отчета истекшего времени *ET*.
  b) В каждом периоде выполнения задачи проверяется, достигло ли время *ET* значения *PT*. Если время *ET* достигает значения *PT* сразу после выполнения данной проверки, это приводит к задержке длительностью в один период задачи.
- В Sysmac Studio время отображается с шагом 0,001 мс, но точность отсчета времени составляет 1 нс.
- Если на входе *In* уже присутствует состояние ИСТИНА, таймер запускается, как только начинается работа.
- Если для параметра *PT* задано значение T#0 мс или отрицательное число, выход *Q* не перейдет в состояние ИСТИНА, даже если значение *In* поменяется на ИСТИНА.
- Значение *PT* можно изменять, когда вход *In* находится в состоянии ИСТИНА. Команда работает следующим образом:

Состояние таймера	Значение *Q*	Значение *PT* после изменения	Работа
После завершения отсчета времени	ЛОЖЬ	---	*Q* остается в состоянии ЛОЖЬ. Значение *ET* также не изменяется. Оно остается равным значению *PT* до изменения.
Выполняется отсчет времени	ИСТИНА	PT ≥ ET	Отсчет времени продолжается. Когда значение *ET* достигает значения *PT*, значение *Q* меняется на ЛОЖЬ и *ET* больше не увеличивается.
		PT < ET	Значение *Q* немедленно меняется на ЛОЖЬ. Увеличение *ET* немедленно прекращается.

- Если эта команда находится в области главного управления и область главного управления сброшена, отсчет времени продолжается до конца, если таймер работает. Затем значение *ET* становится равно 0, а значение *Q* меняется на ЛОЖЬ. Однако если к выходу *Q* подключена команда Out, условие выполнения команды Out остается равным ЛОЖЬ, даже если на выходе *Q* присутствует состояние ИСТИНА.
- Если эта команда не выполняется из-за выполнения какой-либо команды перехода (например, команды JMP), значение *ET* не обновляется и отсчет времени не производится. Когда эта команда выполняется снова, отсчет времени перезапускается.
- При использовании этой команды в лестничной диаграмме значение выхода *Q* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

AccumulationTimer

Команда AccumulationTimer измеряет общее время, в течение которого вход таймера находился в состоянии «ИСТИНА».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Accumulation Timer	Накопительный таймер	FB		AccumulationTimer_instance(In, PT, Reset, Q, ET);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Вход таймера	Вход	ИСТИНА: таймер работает ЛОЖЬ: Таймер остановлен	Зависит от типа данных.	---	ЛОЖЬ
PT	Установленное время		Максимальное время	*1	мс	0
Reset	Сброс		ИСТИНА: сброс таймера ЛОЖЬ: таймер не сброшен	Зависит от типа данных.	---	ЛОЖЬ
Q	Выход таймера	Выход	ИСТИНА: *ET* достигло значения *PT*. ЛОЖЬ: *ET* не достигло значения *PT*.	Зависит от типа данных.	---	---
ET	Общее время		Общее время	*1	мс	

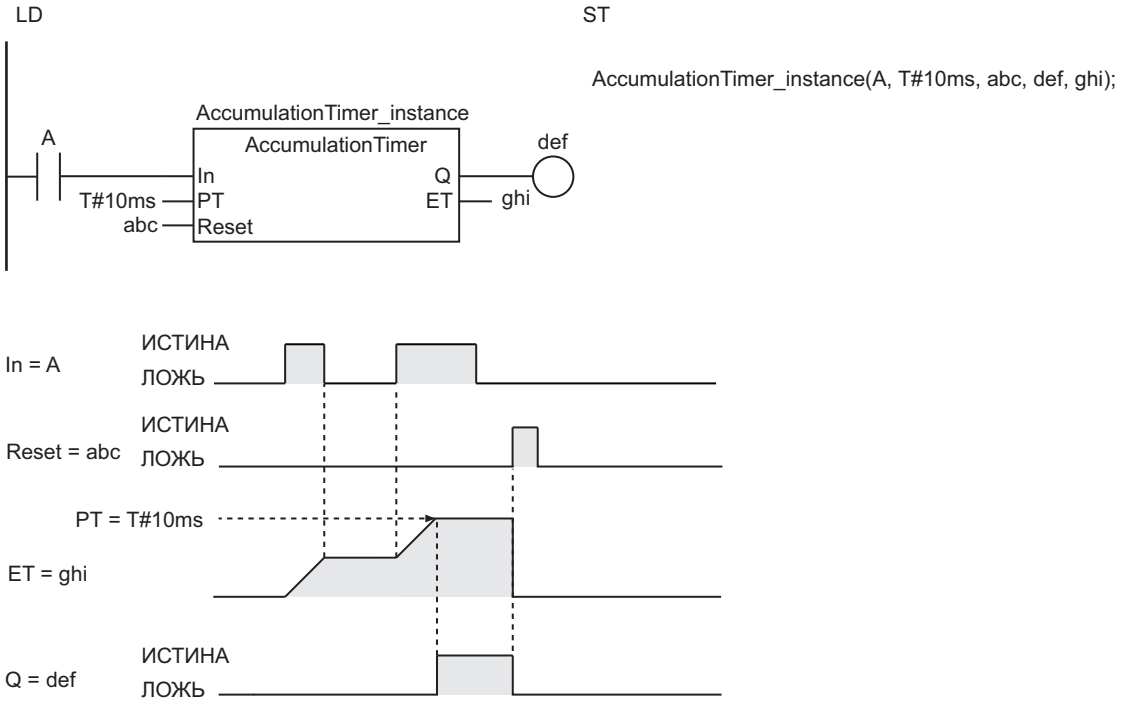
*1. T#0 мс...T#106751д_23ч_47м_16с_854,775807мс

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	ОК																			
PT																ОК				
Reset	ОК																			
Q	ОК																			
ET																ОК				

Функция

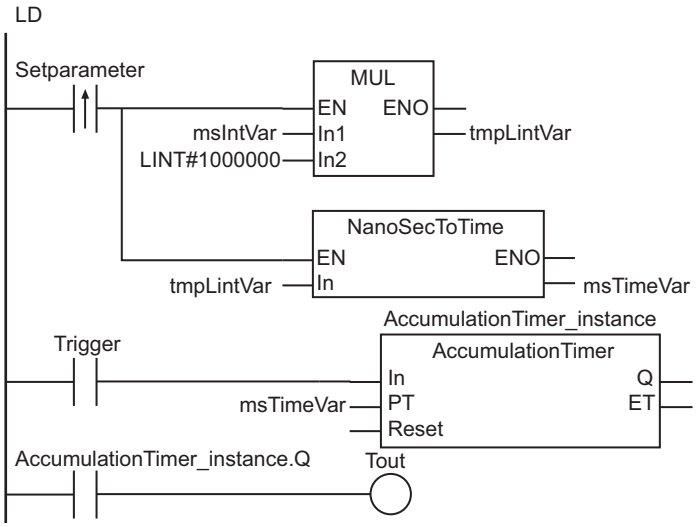
Команда AccumulationTimer измеряет общее время, в течение которого вход таймера *In* находился в состоянии ИСТИНА. Время задается в наносекундах.
Если вход *Reset* находится в состоянии ЛОЖЬ, таймер начинает работать, когда вход *In* переходит в состояние ИСТИНА. По мере отсчета времени значение переменной *ET* (истекшее время) увеличивается.
Если вход *In* переходит в состояние ЛОЖЬ, таймер останавливается. В *ET* при этом сохраняется текущее значение.
Если вход *In* вновь переходит в состояние ИСТИНА, таймер снова запускается. Рост значения переменной *ET* возобновляется с ее текущего значения.
Когда значение *ET* достигает заданного времени *PT*, выход таймера *Q* переходит в состояние ИСТИНА. После этого *ET* больше не увеличивается.
Когда вход *Reset* устанавливается в состояние ИСТИНА, таймер сбрасывается. Значение *ET* становится равно 0, а значение *Q* меняется на ЛОЖЬ.

На рисунке ниже показаны пример программы и временная диаграмма для случая, когда *PT* = T#10 мс. Переменная *abc* принимает значение ИСТИНА, когда общее время достигает 10 мс — общий период времени, в течение которого переменная *A* находилась в состоянии ИСТИНА.



Дополнительная информация

- Если требуется таймер, который сбрасывает свой выход и значение истекшего времени, когда его вход *In* переходит в состояние ЛОЖЬ, используйте команду *TON* на стр. 2-148.
- В случае подключения к устройству HMI, которое не поддерживает тип данных TIME, перед вводом заданного целочисленного значения времени в эту команду его необходимо преобразовать в тип данных TIME. Для преобразования целочисленного значения в значение типа TIME используйте команду *NanoSecToTime* на стр. 2-727. Для преобразования значения типа TIME в целочисленное значение используйте команду *TimeToNanoSec* на стр. 2-723. В обеих командах время выражается в наносекундах. В представленном ниже примере программы переменная *msIntVar* типа INT — это установленное время в миллисекундах.



ST

```
tmpLintVar:=msIntVar*LINT#1000000;
msTimeVar:=NanoSecToTime(tmpLintVar);
AccumulationTimer_instance(In:=Trigger, PT:=msTimeVar, Q=>Tout);
```

Меры предосторожности для обеспечения надлежащей эксплуатации

- Ошибка отсчета времени, при которой выход *Q* = ИСТИНА для *PT*: от -100 нс до (100 нс + 1 период задачи).
  Вышеуказанный диапазон включает следующее:
  a) ±100 нс — ошибка отчета истекшего времени *ET*.
  b) В каждом периоде выполнения задачи проверяется, достигло ли время *ET* значения *PT*. Если время *ET* достигает значения *PT* сразу после выполнения данной проверки, это приводит к задержке длительностью в один период задачи.
- В Sysmac Studio время отображается с шагом 0,001 мс, но точность отсчета времени составляет 1 нс.
- Если входы *In* и *Reset* оба находятся в состоянии ИСТИНА, приоритетом обладает вход *Reset*. То есть *ET* сбрасывается в 0, а выход *Q* сбрасывается в ЛОЖЬ.
- Если на входе *In* уже присутствует состояние ИСТИНА, таймер запускается, как только начинается работа.
- Если для параметра *PT* задано значение T#0 мс или отрицательное число, выход *Q* перейдет в состояние ИСТИНА сразу, как только значение *In* поменяется на ИСТИНА.
- Значение *PT* можно изменить до того, как переменная *ET* достигнет значения *PT*. Команда работает следующим образом:

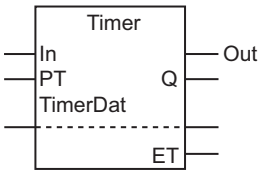
Состояние таймера	Значение *Q*	Значение *PT* после изменения	Работа
После завершения отсчета времени	ИСТИНА	---	*Q* остается в состоянии ИСТИНА. Значение *ET* также не изменяется. Оно остается равным значению *PT* до изменения.

Состояние таймера	Значение *Q*	Значение *PT* после изменения	Работа
Выполняется отсчет времени	ЛОЖЬ	PT ≥ ET	Когда значение *In* меняется на ИСТИНА, отсчет времени продолжается. Когда значение *ET* достигает значения *PT*, значение *Q* меняется на ИСТИНА и *ET* больше не увеличивается.
		PT < ET	Когда значение *In* меняется на ИСТИНА, значение *Q* немедленно меняется на ИСТИНА. Увеличение *ET* немедленно прекращается.

- Если эта команда находится в области главного управления и область главного управления сброшена, команда работает следующим образом:
  a) Таймер останавливается. Текущие значения *ET* и *Q* при этом сохраняются.
  b) После отмены сброса главного управления увеличение *ET* возобновляется с сохраненного значения.
  c) Если к выходу *Q* подключена команда Out, условие выполнения команды Out остается равным ЛОЖЬ, даже если на выходе *Q* присутствует состояние ИСТИНА.
  d) Активируется вход *Reset*.
- Если эта команда не выполняется из-за выполнения какой-либо команды перехода (например, команды JMP), значение *ET* не обновляется. Однако отсчет времени по-прежнему продолжается. Поэтому при следующем выполнении данной команды переменная *ET* принимает правильное значение.
- При использовании этой команды в лестничной диаграмме значение выхода *Q* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

Timer

Команда Timer выдает состояние «ИСТИНА» по истечении заданного времени с момента запуска таймера. Время задается с шагом 100 мс.

Команда	Назначение	FB/ FUN	Графическое представление	Выражение языка ST
Timer	100 мс таймер	FUN		Out:=Timer(In, PT, TimerDat, Q, ET);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Вход таймера	Вход	ИСТИНА: сигнал запуска таймера ЛОЖЬ: сигнал сброса таймера	Зависит от типа данных.	---	ЛОЖЬ
PT	Установленное время		Время с момента запуска таймера, по истечении которого выход *Q* переходит в состояние «ИСТИНА».		мс	*1
TimerDat	Состояние таймера	Вход-выход	Текущее состояние таймера	---	---	---
Out	Возвращаемое значение	Выход	ИСТИНА: выход таймера включен ЛОЖЬ: выход таймера выключен	Зависит от типа данных.	---	---
Q	Выход таймера		Такое же назначение, как у *Out*.			
ET	Оставшееся время		Оставшееся время		мс	

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	ОК																			
PT							ОК													
TimerDat	Structure _sTimer																			
Out	ОК																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Q	OK																			
ET							OK													

Функция

Команда Timer выдает состояние «ИСТИНА» по истечении заданного времени с момента запуска таймера. Время задается с шагом 100 мс.

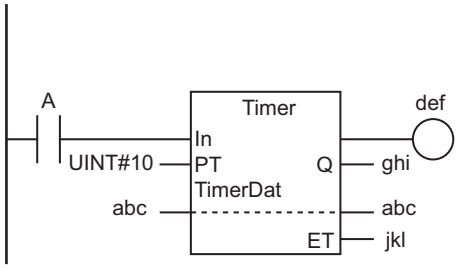
Когда вход таймера *In* переходит в состояние ЛОЖЬ, таймер сбрасывается. Оставшееся время (*ET*) становится равно уставке таймера (*PT*), а выход таймера (*Q*) переходит в состояние ЛОЖЬ. Таймер запускается, когда вход *In* переходит в состояние ИСТИНА. По мере отсчета времени значение переменной *ET* (истекшее время) уменьшается.
Когда значение *ET* достигает 0, выход таймера *Q* переходит в состояние ИСТИНА. После этого *ET* больше не уменьшается.

Если вход *In* переходит в состояние ЛОЖЬ после запуска таймера, таймер сбрасывается, даже если *ET* еще не достигло 0.

Тип данных переменной состояния таймера *TimerDat* — структура _sTimer.

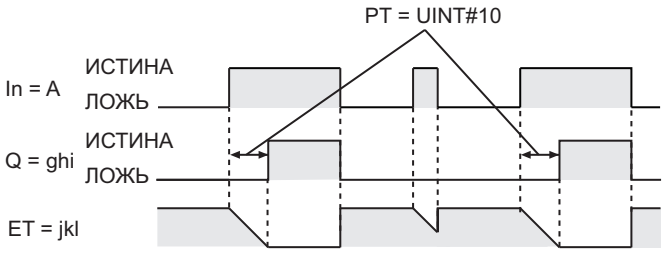
На рисунке ниже показаны пример программы и временная диаграмма для случая, когда *PT* = UINT#10. Переменная *ghi* примет значение ИСТИНА через 1000 мс (1 с) после того, как значение переменной *A* поменяется на ИСТИНА.

LD



ST

```
def:=Timer(A, UINT#10, abc, ghi, jkl);
```



Дополнительная информация

Для более точного измерения времени следует использовать команду *TON* на стр. 2-148, которая измеряет время в наносекундах. Команда TON при выполнении измеряет время в наносекундах, поэтому она точнее команды Timer. С другой стороны, время выполнения команды Timer меньше.

Меры предосторожности для обеспечения надлежащей эксплуатации

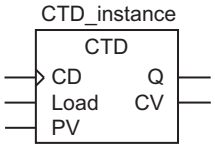
- Отсчет времени начинается в начале работы программного компонента, содержащего эту команду. Следовательно, значение *ET* будет одинаковым независимо от того, в каком месте программного компонента выполняется команда.
- Ошибка отсчета времени, при которой выход *Q* переходит в состояние ИСТИНА для *PT*: + 1 период задачи (задержка, равная одному периоду задачи).
  Вышеуказанный диапазон включает следующее:
  a) В каждом цикле выполнения задачи проверяется, достигло ли истекшее время *ET* заданного значения *PT*. Если время *ET* достигнет значения *PT* сразу после выполнения данной проверки, это приведет к задержке длительностью в один период задачи.
- Хотя *TimerDat* является переменной типа «вход-выход», передавать в нее какие-либо значения не требуется. Создайте область памяти под размер структуры _sTimer и передайте ее команде.
- Не меняйте содержимое *TimerDat*.
- Если на входе *In* присутствует состояние ИСТИНА, таймер запускается, как только начинается работа.
- Если значение *PT* изменяется, новое значение отражается при следующем сбросе таймера. Во время отсчета времени значение не обновляется.
- Если эта команда находится в области главного управления и область главного управления сброшена, таймер сбрасывается. Значение *ET* становится равно значению *PT*, а значение *Q* меняется на ЛОЖЬ.
- Если эта команда не выполняется из-за выполнения какой-либо команды перехода (например, команды JMP), значение *ET* не обновляется. Однако отсчет времени по-прежнему продолжается. Поэтому при следующем выполнении данной команды переменная *ET* принимает правильное значение.
- При использовании этой команды в лестничной диаграмме значения выходов *Q* и *Out* меняются на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

Команды счетчиков

Команда	Имя	Стр.
CTD	Обратный счетчик	стр. 2-170
CTD_**	Групповой обратный счетчик	стр. 2-173
CTU	Прямой счетчик	стр. 2-176
CTU_**	Групповой прямой счетчик	стр. 2-179
CTUD	Прямой-обратный счетчик	стр. 2-182
CTUD_**	Групповой прямой-обратный счетчик	стр. 2-187

CTD

Команда CTD реализует счетчик обратного счета. Текущее значение счетчика уменьшается при поступлении сигнала на вход счетчика. Предустановленное и текущее значения счетчика должны относиться к типу данных INT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CTD	Обратный счетчик	FB		CTD_instance (CD, Load, PV, Q, CV);

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
CD	Вход счетчика	Вход	Вход счетчика	Зависит от типа данных.	---	ЛОЖЬ
Load*1	Сигнал загрузки		ИСТИНА: записать в *CV* значение *PV*.			
PV	Предустановленное значение		Установленное значение счетчика	0…32 767		0
Q	Выход счетчика	Выход	ИСТИНА: выход счетчика включен ЛОЖЬ: выход счетчика выключен	Зависит от типа данных.	---	---
CV	Текущее значение счетчика		Текущее значение счетчика	От 0 до 32767		

*1. В Sysmac Studio версии 1.03 или более поздней версии есть возможность использовать *LD* вместо *Load*. Так нагляднее отображается соответствие между переменными и именами параметров в выражениях языка ST. Например, можно использовать следующую форму записи: CTD_instance(CD:=A, LD:=abc, PV:=INT#5, Q=>def, CV=>ghi);.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CD	ОК																			
Load	ОК																			
PV											ОК									
Q	ОК																			
CV											ОК									

Функция

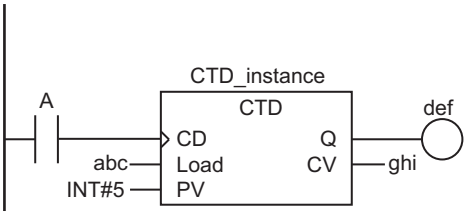
Команда CTD создает обратный счетчик. Предустановленное и текущее значения счетчика должны относиться к типу данных INT.

Когда сигнал загрузки *Load* переходит в состояние ИСТИНА, в текущее значение счетчика *CV* записывается предустановленное значение *PV*, а выход счетчика *Q* сбрасывается в ЛОЖЬ.
Когда входной сигнал счетчика *CD* переходит в состояние ИСТИНА, значение *CV* уменьшается.
Когда значение *CV* становится равно или меньше 0, выход *Q* переходит в состояние ИСТИНА.

После того как *CV* становится равно или меньше 0, *CV* больше не меняется, даже если *CD* переходит в состояние ИСТИНА.
Пока на входе *Load* состояние ИСТИНА, вход *CD* игнорируется. Значение *CV* не уменьшается.

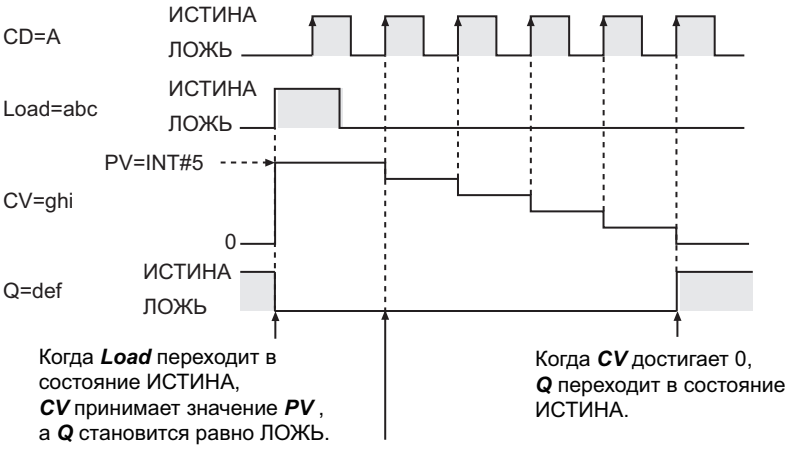
На рисунке ниже показаны пример программы и временная диаграмма для случая, когда *PV* = INT#5.

LD



ST

```
CTD_instance(A, abc, INT#5, def, ghi);
```



Дополнительная информация

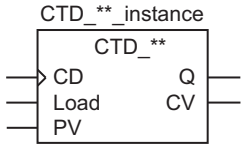
- Для создания счетчика, текущее значение которого увеличивается при каждом поступлении входного сигнала, используйте команду *CTU* на стр. 2-176.
- Для создания счетчика, текущее значение которого может уменьшаться и увеличиваться, используйте команду *CTUD* на стр. 2-182.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Для перезапуска счетчика, который завершил обратный счет, переведите вход *Load* в состояние ИСТИНА, а затем снова в ЛОЖЬ.
- Даже если для *PV* установлено отрицательное значение, *CV* примет значение *PV*, когда значение *Load* поменяется на ИСТИНА. Так как значение *CV* равно или меньше 0, выход *Q* немедленно перейдет в состояние ИСТИНА. После этого значение *CV* не будет уменьшаться, даже если значение *CD* будет меняться.
- Если в момент, когда *CD* находится в состоянии ЛОЖЬ, прервется питание или режим работы поменяется на «Программирование», а при перезапуске данной команды *CD* будет находиться в состоянии ИСТИНА, значение *CV* однократно уменьшится.
- При использовании этой команды в лестничной диаграмме значение выхода *Q* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

CTD_**

Команда CTD_** реализует счетчик обратного счета. Текущее значение счетчика уменьшается при поступлении сигнала на вход счетчика. Предустановленное и текущее значения счетчика должны относиться к одному из следующих типов данных: DINT, LINT, UDINT или ULINT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CTD_**	Групповой обратный счетчик	FB	 "**" должно быть DINT, LINT, UDINT или ULINT.	CTD_**_instance (CD, Load, PV, Q, CV); "**" должно быть DINT, LINT, UDINT или ULINT.

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
CD	Вход счетчика	Вход	Вход счетчика	Зависит от типа данных.	---	ЛОЖЬ
Load[*1]	Сигнал загрузки		ИСТИНА: записать в *CV* значение *PV*.			
PV	Предустановленное значение		Установленное значение счетчика	Зависит от типа данных.[*2]		0
Q	Выход счетчика	Выход	ИСТИНА: выход счетчика включен ЛОЖЬ: выход счетчика выключен	Зависит от типа данных.	---	---
CV	Текущее значение счетчика		Текущее значение счетчика	Зависит от типа данных.[*2]		

*1. В Sysmac Studio версии 1.03 или более поздней версии есть возможность использовать *LD* вместо *Load*. Так нагляднее отображается соответствие между переменными и именами параметров в выражениях языка ST. Например, можно использовать следующую форму записи: CTD_LINT_instance(CD:=A, LD:=abc, PV:=LINT#5, Q=>def, CV=>ghi);.

*2. Отрицательные числа исключены.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CD	OK																			
Load	OK																			
PV								OK	OK			OK	OK							
Q	OK																			
CV	Тип данных должен быть таким же, как у *PV*.																			

Функция

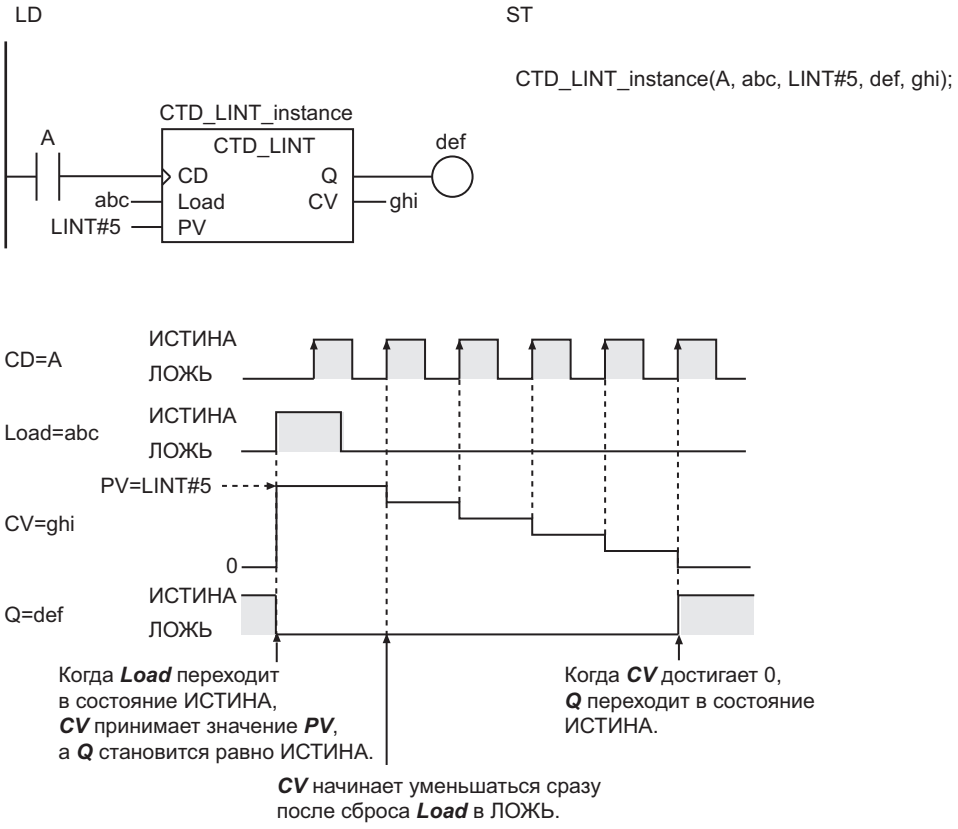
Команда CTD_** создает обратный счетчик. Предустановленное и текущее значения счетчика должны относиться к одному из следующих типов данных: DINT, LINT, UDINT или ULINT. Имя команды определяется типом данных *PV* и *CV*. Например, если используется тип данных INT, команда будет иметь имя CTD_LINT.

Когда сигнал загрузки *Load* переходит в состояние ИСТИНА, в текущее значение счетчика *CV* записывается предустановленное значение *PV*, а выход счетчика *Q* сбрасывается в ЛОЖЬ. Когда входной сигнал счетчика *CD* переходит в состояние ИСТИНА, значение *CV* уменьшается. Когда значение *CV* становится равно или меньше 0, выход *Q* переходит в состояние ИСТИНА.

После того как *CV* становится равно или меньше 0, *CV* больше не меняется, даже если *CD* переходит в состояние ИСТИНА.

Пока на входе *Load* состояние ИСТИНА, вход *CD* игнорируется. Значение *CV* не уменьшается.

На рисунке ниже показаны пример программы и временная диаграмма для команды CTD_LINT, в которой *PV* = LINT#5.



Дополнительная информация

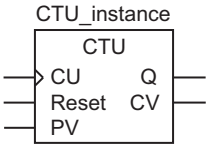
- Для создания счетчика, текущее значение которого увеличивается при каждом поступлении входного сигнала, используйте команду *CTU* на стр. 2-176.
- Для создания счетчика, текущее значение которого может уменьшаться и увеличиваться, используйте команду *CTUD* на стр. 2-182.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Для перезапуска счетчика, который завершил обратный счет, переведите вход *Load* в состояние ИСТИНА, а затем снова в ЛОЖЬ.
- Для *PV* и *CV* следует использовать один и тот же тип данных.
- Даже если для *PV* установлено отрицательное значение, *CV* примет значение *PV*, когда значение *Load* поменяется на ИСТИНА. Так как значение *CV* равно или меньше 0, выход *Q* немедленно перейдет в состояние ИСТИНА. После этого значение *CV* не будет уменьшаться, даже если значение *CD* будет меняться.
- Если в момент, когда *CD* находится в состоянии ЛОЖЬ, прервется питание или режим работы поменяется на «Программирование», а при перезапуске данной команды *CD* будет находиться в состоянии ИСТИНА, значение *CV* однократно уменьшится.
- При использовании этой команды в лестничной диаграмме значение выхода *Q* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

CTU

Команда CTU реализует счетчик прямого счета. Текущее значение счетчика увеличивается при поступлении сигнала на вход счетчика. Предустановленное и текущее значения счетчика должны относиться к типу данных INT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CTU	Прямой счетчик	FB		CTU_instance (CU, Reset, PV, Q, CV);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
CU	Вход счетчика	Вход	Вход счетчика	Зависит от типа данных.	---	ЛОЖЬ
Reset*1	Сигнал сброса		ИСТИНА: сброс *CV* в 0.			
PV	Предустановленное значение		Установленное значение счетчика	От 0 до 32767		0
Q	Выход счетчика	Выход	ИСТИНА: выход счетчика включен ЛОЖЬ: выход счетчика выключен	Зависит от типа данных.	---	---
CV	Текущее значение счетчика		Текущее значение счетчика	От 0 до 32767		

*1. В Sysmac Studio версии 1.03 или более поздней версии есть возможность использовать *R* вместо *Reset*. Так нагляднее отображается соответствие между переменными и именами параметров в выражениях языка ST. Например, можно использовать следующую форму записи: CTU_instance(CU:=A, R:=abc, PV:=INT#5, Q=>def, CV=>ghi);.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	ОК																			
Reset	ОК																			
PV											ОК									
Q	ОК																			
CV											ОК									

Функция

Команда CTU создает прямой счетчик. Предустановленное и текущее значения счетчика должны относиться к типу данных INT.

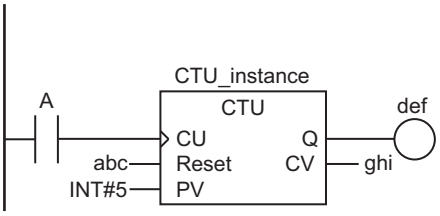
Когда сигнал сброса *Reset* переходит в состояние ИСТИНА, текущее значение счетчика *CV* становится равно 0, а выход счетчика *Q* сбрасывается в ЛОЖЬ.
Когда входной сигнал счетчика *CU* переходит в состояние ИСТИНА, значение *CV* увеличивается. Когда значение *CV* становится равно или больше *PV*, выход *Q* переходит в состояние ИСТИНА.

После того как значение *CV* становится равно или больше *PV*, значение *CV* больше не меняется, даже если *CU* переходит в состояние ИСТИНА.

Пока на входе *Reset* состояние ИСТИНА, вход *CU* игнорируется. Значение *CV* не увеличивается.

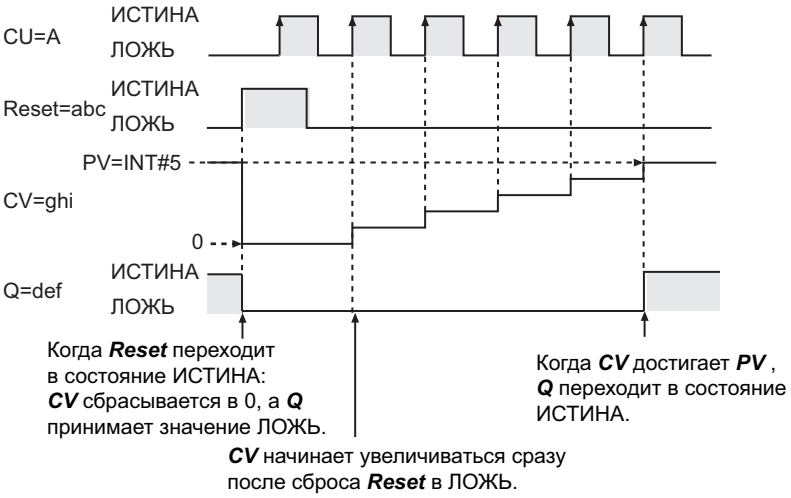
На рисунке ниже показаны пример программы и временная диаграмма для случая, когда *PV* = INT#5.

LD



ST

```
CTU_instance(A, abc, INT#5, def, ghi);
```



Дополнительная информация

- Для создания счетчика, текущее значение которого уменьшается при каждом поступлении входного сигнала, используйте команду *CTD* на стр. 2-170.
- Для создания счетчика, текущее значение которого может уменьшаться и увеличиваться, используйте команду *CTUD* на стр. 2-182.

Меры предосторожности для обеспечения надлежащей эксплуатации

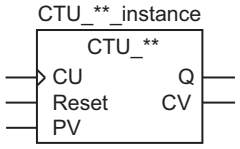
- Для перезапуска счетчика, который завершил прямой счет, переведите вход *Reset* в состояние ИСТИНА, а затем снова в ЛОЖЬ.
- Даже если для *PV* установлено отрицательное значение, *CV* примет значение 0, когда значение на входе *Reset* поменяется на ИСТИНА. Так как значение *CV* при этом будет больше значения *PV*, выход *Q* немедленно перейдет в состояние ИСТИНА. После этого значение *CV* не будет увеличиваться, даже если значение *CU* будет меняться.
- Если значение *PV* изменяется, когда вход *Reset* находится в состоянии ЛОЖЬ, команда работает следующим образом:

Значение *PV*	Значение
Больше, чем текущее значение *CV*	Операция счета продолжается.
Равно или меньше текущего значения *CV*	Операция счета прекращается. Значение *Q* меняется на ИСТИНА. Текущее значение *CV* при этом сохраняется и не изменяется.

- Если в момент, когда *CU* находится в состоянии ЛОЖЬ, прервется питание или режим работы поменяется на «Программирование», а при перезапуске данной команды *CU* будет находиться в состоянии ИСТИНА, значение *CV* однократно увеличится.
- При использовании этой команды в лестничной диаграмме значение выхода *Q* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

CTU_**

Команда CTU_** реализует счетчик прямого счета. Текущее значение счетчика увеличивается при поступлении сигнала на вход счетчика. Предустановленное и текущее значения счетчика должны относиться к одному из следующих типов данных: DINT, LINT, UDINT или ULINT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CTU_**	Групповой прямой счетчик	FB	 "**" должно быть DINT, LINT, UDINT или ULINT.	CTU_**_instance (CU, Reset, PV, Q, CV); "**" должно быть DINT, LINT, UDINT или ULINT.

Переменные

	Назначение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
CU	Вход счетчика	Вход	Вход счетчика	Зависит от типа данных.	---	ЛОЖЬ
Reset[*1]	Сигнал сброса		ИСТИНА: сброс *CV* в 0.			
PV	Предустановленное значение		Установленное значение счетчика	Зависит от типа данных.[*2]		0
Q	Выход счетчика	Выход	ИСТИНА: выход счетчика включен ЛОЖЬ: выход счетчика выключен	Зависит от типа данных.	---	---
CV	Текущее значение счетчика		Текущее значение счетчика	Зависит от типа данных.[*2]		

*1. В Sysmac Studio версии 1.03 или более поздней версии есть возможность использовать *R* вместо *Reset*. Так нагляднее отображается соответствие между переменными и именами параметров в выражениях языка ST. Например, можно использовать следующую форму записи: CTU_LINT_instance(CU:=A, R:=abc, PV:=LINT#5, Q=>def, CV=>ghi);.

*2. Отрицательные числа исключены.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	OK																			
Reset	OK																			
PV								OK	OK			OK	OK							
Q	OK																			
CV	Тип данных должен быть таким же, как у *PV*.																			

Функция

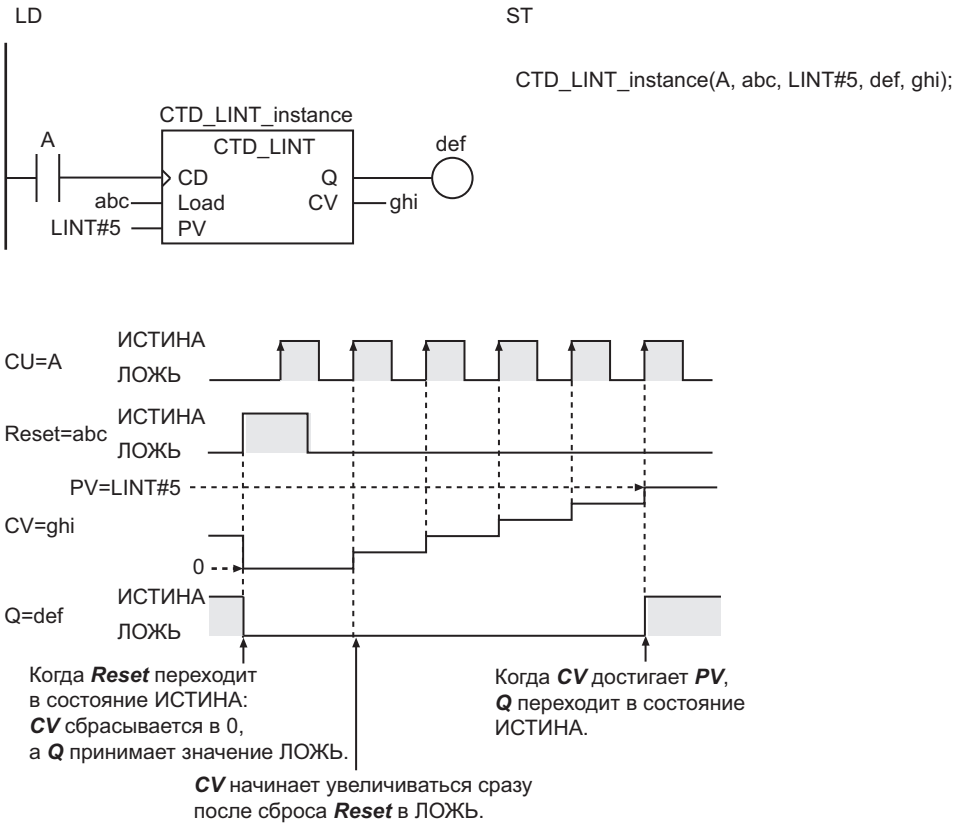
Команда CTU_** создает прямой счетчик. Предустановленное и текущее значения счетчика должны относиться к одному из следующих типов данных: DINT, LINT, UDINT или ULINT.
Имя команды определяется типом данных *PV* и *CV*. Например, если используется тип данных LINT, команда будет иметь имя CTU_LINT.

Когда сигнал сброса *Reset* переходит в состояние ИСТИНА, текущее значение счетчика *CV* становится равно 0, а выход счетчика *Q* сбрасывается в ЛОЖЬ.
Когда входной сигнал счетчика *CU* переходит в состояние ИСТИНА, значение *CV* увеличивается. Когда значение *CV* становится равно или больше *PV*, выход *Q* переходит в состояние ИСТИНА.

После того как значение *CV* становится равно или больше *PV*, значение *CV* больше не меняется, даже если *CU* переходит в состояние ИСТИНА.

Пока на входе *Reset* состояние ИСТИНА, вход *CU* игнорируется. Значение *CV* не увеличивается.

На рисунке ниже показаны пример программы и временная диаграмма для команды CTU_LINT, в которой *PV* = LINT#5.



Дополнительная информация

- Для создания счетчика, текущее значение которого уменьшается при каждом поступлении входного сигнала, используйте команду *CTD* на стр. 2-170.
- Для создания счетчика, текущее значение которого может уменьшаться и увеличиваться, используйте команду *CTUD* на стр. 2-182.

Меры предосторожности для обеспечения надлежащей эксплуатации

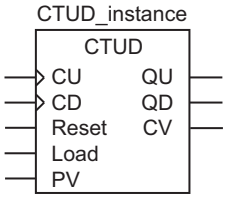
- Для перезапуска счетчика, который завершил прямой счет, переведите вход *Reset* в состояние ИСТИНА, а затем снова в ЛОЖЬ.
- Даже если для *PV* установлено отрицательное значение, *CV* примет значение 0, когда значение на входе *Reset* поменяется на ИСТИНА. Так как значение *CV* при этом будет больше значения *PV*, выход *Q* немедленно перейдет в состояние ИСТИНА. После этого значение *CV* не будет увеличиваться, даже если значение *CU* будет меняться.
- Для *PV* и *CV* следует использовать один и тот же тип данных.
- Если значение *PV* изменяется, когда вход *Reset* находится в состоянии ЛОЖЬ, команда работает следующим образом:

Значение *PV*	Значение
Больше, чем текущее значение *CV*	Операция счета продолжается.
Равно или меньше текущего значения *CV*	Операция счета прекращается. Значение *Q* меняется на ИСТИНА. Текущее значение *CV* при этом сохраняется и не изменяется.

- Если в момент, когда *CU* находится в состоянии ЛОЖЬ, прервется питание или режим работы поменяется на «Программирование», а при перезапуске данной команды *CU* будет находиться в состоянии ИСТИНА, значение *CV* однократно увеличится.
- При использовании этой команды в лестничной диаграмме значение выхода *Q* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

CTUD

Команда CTUD реализует счетчик, ведущий счет в прямом и обратном направлениях в соответствии с сигналами на входах прямого и обратного счета. Предустановленное и текущее значения счетчика должны относиться к типу данных INT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CTUD	Прямой-обратный счетчик	FB		CTUD_instance (CU, CD, Reset, Load, PV, QU, QD, CV);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
CU	Вход прямого счета	Вход	Вход прямого счета	Зависит от типа данных.	---	ЛОЖЬ
CD	Вход обратного счета		Вход обратного счета			
Reset*1	Сигнал сброса		ИСТИНА: сброс *CV* в 0.			
Load*1	Сигнал загрузки		ИСТИНА: записать в *CV* значение *PV*.			
PV	Предустановленное значение		Конечное значение счетчика при работе в качестве счетчика прямого счета. Начальное значение счетчика при работе в качестве счетчика обратного счета.	От 0 до 32767		0
QU	Выход прямого счета	Выход	ИСТИНА: выход прямого счета включен ЛОЖЬ: выход прямого счета выключен	Зависит от типа данных.	---	---
QD	Выход обратного счета		ИСТИНА: выход обратного счета включен ЛОЖЬ: выход обратного счета выключен			
CV	Текущее значение счетчика		Текущее значение счетчика	От 0 до 32767		

*1. В Sysmac Studio версии 1.03 или более поздней версии есть возможность использовать *R* вместо *Reset* и *LD* вместо *Load*. Так нагляднее отображается соответствие между переменными и именами параметров в выражениях языка ST.
Например, можно использовать следующую форму записи: CTUD_instance(CU:=A, CD:=B, R:=abc, LD:=def, PV:=INT#3, QU=>ghi, QD=>jkl, CV=>mno);.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	OK																			
CD	OK																			
Reset	OK																			
Load	OK																			
PV											OK									
QU	OK																			
QD	OK																			
CV											OK									

Функция

Команда CTUD реализует счетчик, ведущий счет в прямом и обратном направлениях в соответствии с сигналами на входах прямого и обратного счета.
Она функционирует одновременно как прямой счетчик и как обратный счетчик.
Предустановленное и текущее значения счетчика должны относиться к типу данных INT.

Работа в качестве прямого счетчика

Когда сигнал сброса *Reset* переходит в состояние ИСТИНА, текущее значение счетчика *CV* становится равно 0, а выход прямого счета *QU* сбрасывается в ЛОЖЬ.
Когда входной сигнал прямого счета *CU* переходит в состояние ИСТИНА, значение *CV* увеличивается. Когда значение *CV* становится равно или больше *PV*, выход *QU* переходит в состояние ИСТИНА.
После того как значение *CV* становится равно или больше *PV*, значение *CV* больше не меняется, даже если *CU* переходит в состояние ИСТИНА.

Работа в качестве обратного счетчика

Когда сигнал загрузки *Load* переходит в состояние ИСТИНА, в текущее значение счетчика *CV* записывается предустановленное значение *PV*, а выход обратного счета *QD* сбрасывается в ЛОЖЬ.
Когда входной сигнал обратного счета *CD* переходит в состояние ИСТИНА, значение *CV* уменьшается. Когда значение *CV* становится равно или меньше 0, выход *QD* переходит в состояние ИСТИНА.
После того как *CV* становится равно или меньше 0, *CV* больше не меняется, даже если *CD* переходит в состояние ИСТИНА.

Правила работы, общие для прямого и обратного счетчиков

Когда входы *Load* и *Reset* находятся в состоянии ИСТИНА, входы *CU* и *CD* игнорируются. Значение *CV* при этом не увеличивается и не уменьшается.

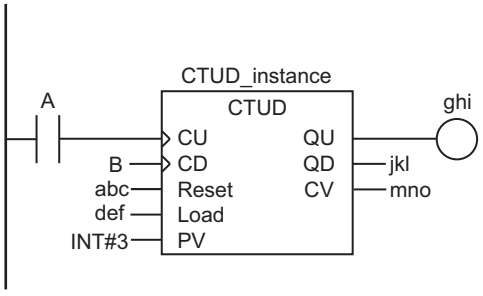
Если входы *CU* и *CD* переходят в состояние ИСТИНА одновременно, значение *CV* не изменяется.
Если оба входа *Reset* и *Load* находятся в состоянии ИСТИНА, вход *Reset* обладает приоритетом и значение *CV* меняется на 0.
Если вход *Reset* переходит в состояние ИСТИНА, то *CV* сбрасывается в 0, поэтому на выходе *QD* устанавливается состояние ИСТИНА.
Если вход *Load* переходит в состояние ИСТИНА, то *CV* принимает значение *PV*, поэтому на выходе *QU* устанавливается состояние ИСТИНА.

В следующей таблице отражена взаимосвязь между *Reset*, *Load*, *CV*, *QU* и *QD*. Предполагается, что значение *PV* больше 0.

Reset	Load	CV	QU	QD	Работа
ЛОЖЬ	ЛОЖЬ	0 или меньше	ЛОЖЬ	ИСТИНА	Работает только прямой счетчик. • Значение *CV* увеличивается, когда *CU* переходит в состояние ИСТИНА. При переходе *CD* в состояние ИСТИНА оно не уменьшается.
		Между 0 и *PV*	ЛОЖЬ	ЛОЖЬ	Работают оба счетчика, прямой и обратный. • Значение *CV* увеличивается, когда *CU* переходит в состояние ИСТИНА, и уменьшается, когда *CD* переходит в состояние ИСТИНА.
		PV или больше	ИСТИНА	ЛОЖЬ	Работает только обратный счетчик. • Значение *CV* уменьшается, когда *CD* переходит в состояние ИСТИНА. При переходе *CU* в состояние ИСТИНА оно не увеличивается.
ИСТИНА	ЛОЖЬ	0	ЛОЖЬ	ИСТИНА	Прямой счетчик сбрасывается. • Значение *CV* сбрасывается в 0.
ЛОЖЬ	ИСТИНА	*PV*	ИСТИНА	ЛОЖЬ	Обратный счетчик сбрасывается. • Значение *CV* становится равно значению *PV*.
ИСТИНА	ИСТИНА	0	ЛОЖЬ	ИСТИНА	Прямой счетчик сбрасывается. Сигнал *Reset* обладает приоритетом над сигналом *Load*. • Значение *CV* сбрасывается в 0.

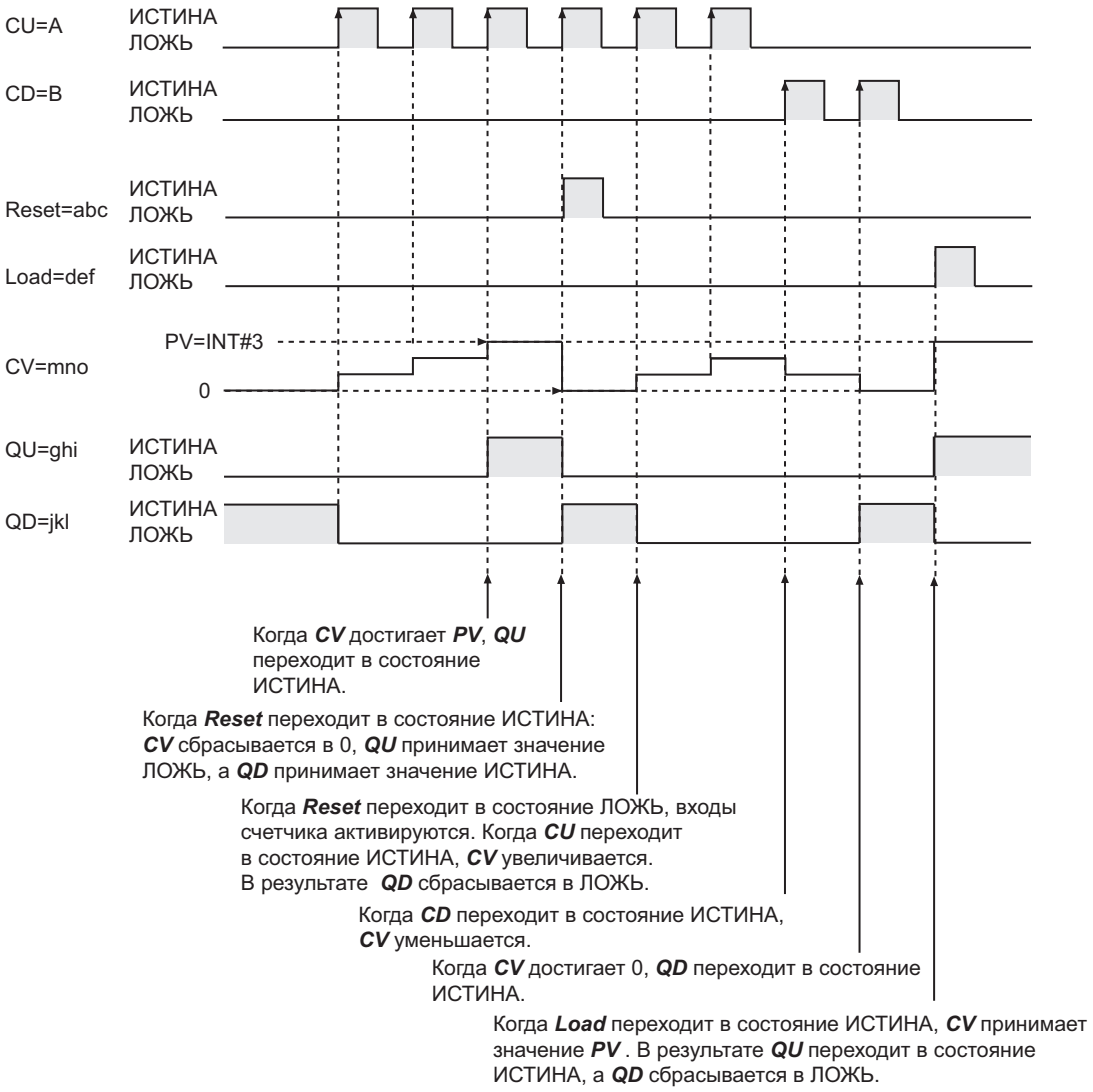
На рисунке ниже показаны пример программы и временная диаграмма для случая, когда *PV* = INT#3.

LD



ST

```
CTUD_instance(A, B, abc, def, INT#3, ghi, jkl, mno);
```



Дополнительная информация

Для создания только обратного или только прямого счетчика используйте, соответственно, команду *CTD* на стр. 2-170 или *CTU* на стр. 2-176.

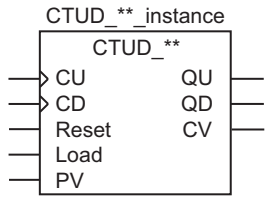
Меры предосторожности для обеспечения надлежащей эксплуатации

- При переводе входа *Reset* в состояние ИСТИНА для сброса прямого счетчика выход *QU* перейдет в состояние ЛОЖЬ, а выход *QD* — в состояние ИСТИНА.
- При переводе входа *Load* в состояние ИСТИНА для сброса обратного счетчика выход *QD* перейдет в состояние ЛОЖЬ, а выход *QU* — в состояние ИСТИНА.
- Даже если для *PV* установлено отрицательное значение, *CV* примет значение *PV*, когда значение *Load* поменяется на ИСТИНА. Так как значение *CV* равно или меньше 0, выход *QD* немедленно перейдет в состояние ИСТИНА. После этого значение *CV* не будет уменьшаться, даже если значение *CD* будет меняться. Когда вход *Reset* переходит в состояние ИСТИНА, значение *CV* меняется на 0. Так как значение *CV* при этом будет равно или больше значения *PV*, выход *QU* немедленно перейдет в состояние ИСТИНА. После этого значение *CV* не будет увеличиваться, даже если значение *CU* будет меняться.

- Значение *PV* можно изменять во время выполнения команды. Если новое значение *PV* будет меньше текущего значения *CV*, выход *QU* немедленно перейдет в состояние ИСТИНА.
- Если в момент, когда *CU* или *CD* находится в состоянии ЛОЖЬ, прервется питание или режим работы поменяется на «Программирование», а при перезапуске данной команды *CU* или *CD* будет находиться в состоянии ИСТИНА, значение *CV*, соответственно, однократно увеличится или уменьшится.

CTUD_**

Команда CTUD_** реализует счетчик, ведущий счет в прямом и обратном направлениях в соответствии с сигналами на входах прямого и обратного счета. Предустановленное и текущее значения счетчика должны относиться к одному из следующих типов данных: DINT, LINT, UDINT или ULINT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CTUD_**	Групповой прямой-обратный счетчик	FB	 "**" должно быть DINT, LINT, UDINT или ULINT.	CTUD_**_instance (CU, CD, Reset, Load, PV, QU, QD, CV); "**" должно быть DINT, LINT, UDINT или ULINT.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
CU	Вход прямого счета	Вход	Вход прямого счета	Зависит от типа данных.	---	ЛОЖЬ
CD	Вход обратного счета		Вход обратного счета			
Reset*1	Сигнал сброса		ИСТИНА: сброс *CV* в 0.			
Load*1	Сигнал загрузки		ИСТИНА: записать в *CV* значение *PV*.			
PV	Предустановленное значение		Конечное значение счетчика при работе в качестве счетчика прямого счета. Начальное значение счетчика при работе в качестве счетчика обратного счета.	Зависит от типа данных.*2		0
QU	Выход прямого счета	Выход	ИСТИНА: выход прямого счета включен ЛОЖЬ: выход прямого счета выключен	Зависит от типа данных.	---	---
QD	Выход обратного счета		ИСТИНА: выход обратного счета включен ЛОЖЬ: выход обратного счета выключен			
CV	Текущее значение счетчика		Текущее значение счетчика	Зависит от типа данных.*2		

*1. В Sysmac Studio версии 1.03 или более поздней версии есть возможность использовать *R* вместо *Reset* и *LD* вместо *Load*. Так нагляднее отображается соответствие между переменными и именами параметров в выражениях языка ST.
Например, можно использовать следующую форму записи: CTUD_LINT_instance(CU:=A, CD:=B, R:=abc, LD:=def, PV:=LINT#3, QU=>ghi, QD=>jkl, CV=>mno);.

*2. Отрицательные числа исключены.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
CU	ОК																			
CD	ОК																			
Reset	ОК																			
Load	ОК																			
PV								ОК	ОК			ОК	ОК							
QU	ОК																			
QD	ОК																			
CV	Тип данных должен быть таким же, как у *PV*.																			

Функция

Команда CTUD_** реализует счетчик, ведущий счет в прямом и обратном направлениях в соответствии с сигналами на входах прямого и обратного счета.
Этот счетчик функционирует одновременно как прямой счетчик и как обратный счетчик.
Предустановленное и текущее значения счетчика должны относиться к одному из следующих типов данных: DINT, LINT, UDINT или ULINT.
Имя команды определяется типом данных *PV* и *CV*. Например, если используется тип данных LINT, команда будет иметь имя CTUD_LINT.

Работа в качестве прямого счетчика

Когда сигнал сброса *Reset* переходит в состояние ИСТИНА, текущее значение счетчика *CV* становится равно 0, а выход прямого счета *QU* сбрасывается в ЛОЖЬ.
Когда входной сигнал прямого счета *CU* переходит в состояние ИСТИНА, значение *CV* увеличивается. Когда значение *CV* становится равно или больше *PV*, выход *QU* переходит в состояние ИСТИНА.
После того как значение *CV* становится равно или больше *PV*, значение *CV* больше не меняется, даже если *CU* переходит в состояние ИСТИНА.

Работа в качестве обратного счетчика

Когда сигнал загрузки *Load* переходит в состояние ИСТИНА, в текущее значение счетчика *CV* записывается предустановленное значение *PV*, а выход обратного счета *QD* сбрасывается в ЛОЖЬ.
Когда входной сигнал обратного счета *CD* переходит в состояние ИСТИНА, значение *CV* уменьшается. Когда значение *CV* становится равно или меньше 0, выход *QD* переходит в состояние ИСТИНА.
После того как *CV* становится равно или меньше 0, *CV* больше не меняется, даже если *CD* переходит в состояние ИСТИНА.

Правила работы, общие для прямого и обратного счетчиков

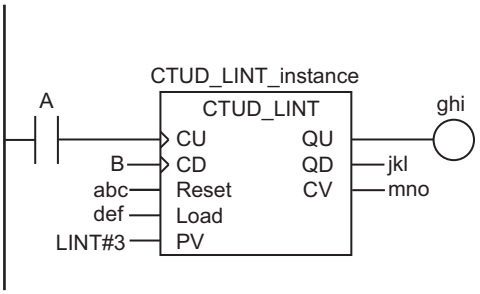
Когда входы *Load* и *Reset* находятся в состоянии ИСТИНА, входы *CU* и *CD* игнорируются. Значение *CV* при этом не увеличивается и не уменьшается.
Если входы *CU* и *CD* переходят в состояние ИСТИНА одновременно, значение *CV* не изменяется.
Если оба входа *Reset* и *Load* находятся в состоянии ИСТИНА, вход *Reset* обладает приоритетом и значение *CV* меняется на 0.
Если вход *Reset* переходит в состояние ИСТИНА, то *CV* сбрасывается в 0, поэтому на выходе *QD* устанавливается состояние ИСТИНА.
Если вход *Load* переходит в состояние ИСТИНА, то *CV* принимает значение *PV*, поэтому на выходе *QU* устанавливается состояние ИСТИНА.

В следующей таблице отражена взаимосвязь между *Reset*, *Load*, *CV*, *QU* и *QD*. Предполагается, что значение *PV* больше 0.

Reset	Load	CV	QU	QD	Работа
ЛОЖЬ	ЛОЖЬ	0 или меньше	ЛОЖЬ	ИСТИНА	Работает только прямой счетчик. • Значение *CV* увеличивается, когда *CU* переходит в состояние ИСТИНА. При переходе *CD* в состояние ИСТИНА оно не уменьшается.
		Между 0 и *PV*	ЛОЖЬ	ЛОЖЬ	Работают оба счетчика, прямой и обратный. • Значение *CV* увеличивается, когда *CU* переходит в состояние ИСТИНА, и уменьшается, когда *CD* переходит в состояние ИСТИНА.
		PV или больше	ИСТИНА	ЛОЖЬ	Работает только обратный счетчик. • Значение *CV* уменьшается, когда *CD* переходит в состояние ИСТИНА. При переходе *CU* в состояние ИСТИНА оно не увеличивается.
ИСТИНА	ЛОЖЬ	0	ЛОЖЬ	ИСТИНА	Прямой счетчик сбрасывается. • Значение *CV* сбрасывается в 0.
ЛОЖЬ	ИСТИНА	*PV*	ИСТИНА	ЛОЖЬ	Обратный счетчик сбрасывается. • Значение *CV* становится равно значению *PV*.
ИСТИНА	ИСТИНА	0	ЛОЖЬ	ИСТИНА	Прямой счетчик сбрасывается. Сигнал *Reset* обладает приоритетом над сигналом *Load*. • Значение *CV* сбрасывается в 0.

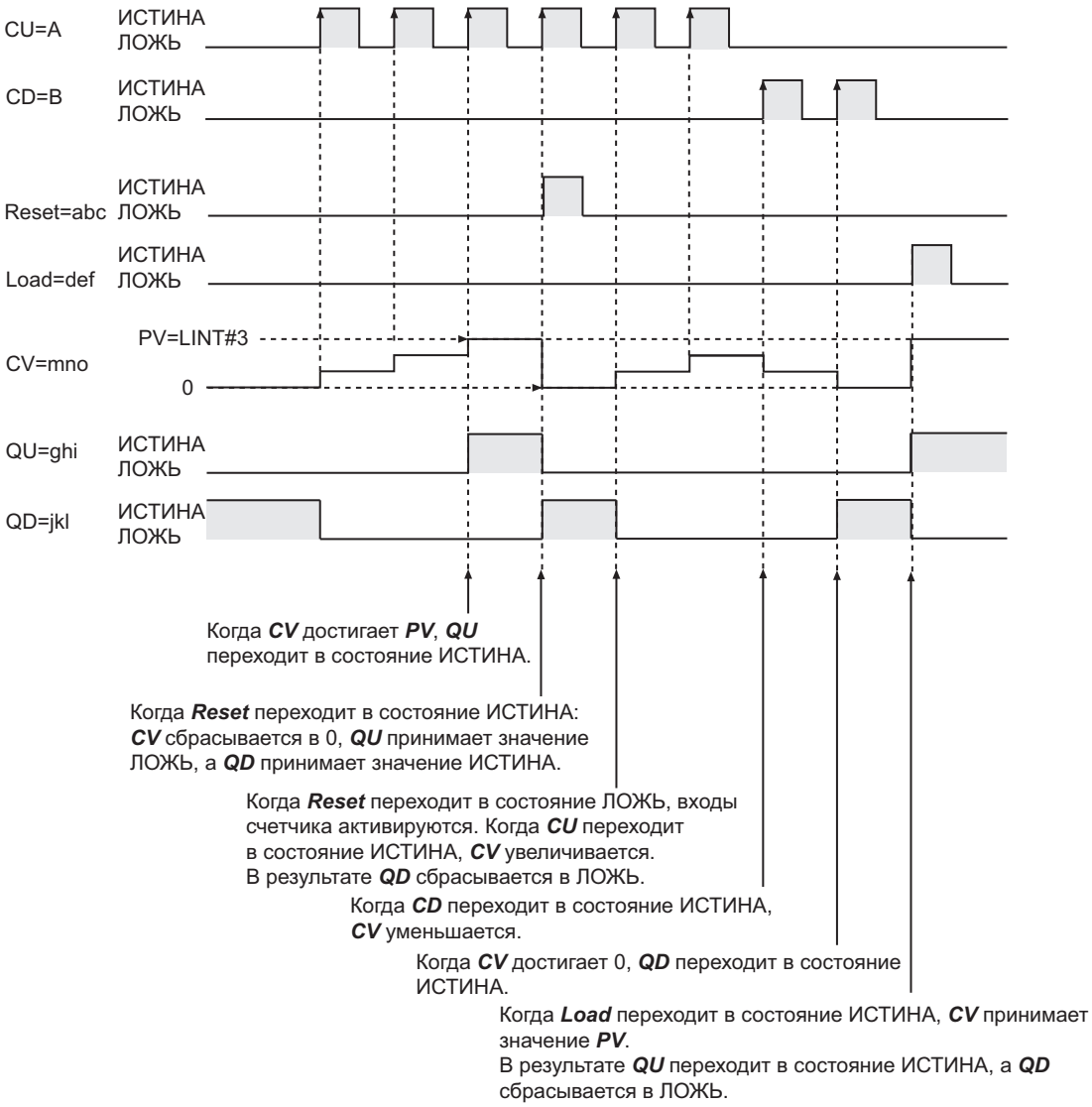
На рисунке ниже показаны пример программы и временная диаграмма для команды CTUD_LINT, в которой *PV* = LINT#3.

LD



ST

```
CTUD_LINT_instance(A, B, abc, def, LINT#3, ghi, jkl, mno);
```



Дополнительная информация

Для создания только обратного или только прямого счетчика используйте, соответственно, команду *CTD_*** на стр. 2-173 или *CTU_*** на стр. 2-179.

Меры предосторожности для обеспечения надлежащей эксплуатации

- При переводе входа *Reset* в состояние ИСТИНА для сброса прямого счетчика выход *QU* перейдет в состояние ЛОЖЬ, а выход *QD* — в состояние ИСТИНА.
- При переводе входа *Load* в состояние ИСТИНА для сброса обратного счетчика выход *QD* перейдет в состояние ЛОЖЬ, а выход *QU* — в состояние ИСТИНА.
- Даже если для *PV* установлено отрицательное значение, *CV* примет значение *PV*, когда значение *Load* поменяется на ИСТИНА. Так как значение *CV* равно или меньше 0, выход *QD* немедленно перейдет в состояние ИСТИНА. После этого значение *CV* не будет уменьшаться, даже если значение *CD* будет меняться. Когда вход *Reset* переходит в состояние ИСТИНА, значение *CV* меняется на 0. Так как значение *CV* при этом будет больше или равно значению

PV, то выход *QU* немедленно перейдет в состояние ИСТИНА. После этого значение *CV* не будет увеличиваться, даже если значение *CU* будет меняться.

- Значение *PV* можно изменять во время выполнения команды. Если новое значение *PV* будет меньше текущего значения *CV*, выход *QU* немедленно перейдет в состояние ИСТИНА.
- Для *PV* и *CV* следует использовать один и тот же тип данных.
- Если в момент, когда *CU* или *CD* находится в состоянии ЛОЖЬ, прервется питание или режим работы поменяется на «Программирование», а при перезапуске данной команды *CU* или *CD* будет находиться в состоянии ИСТИНА, значение *CV*, соответственно, однократно увеличится или уменьшится.

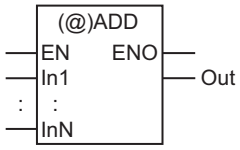
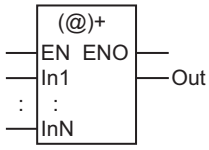
Команды математических операций

Команда	Имя	Стр.
ADD (+)	Сложение	стр. 2-195
AddOU (+OU)	Сложение с проверкой на переполнение	стр. 2-200
SUB (-)	Вычитание	стр. 2-204
SubOU (-OU)	Вычитание с проверкой на переполнение	стр. 2-208
MUL (*)	Умножение	стр. 2-212
MulOU (*OU)	Умножение с проверкой на переполнение	стр. 2-216
DIV (/)	Деление	стр. 2-220
MOD	Деление по модулю	стр. 2-224
ABS	Абсолютное значение	стр. 2-226
RadToDeg и DegToRad	Радианы в градусы/Градусы в радианы	стр. 2-228
SIN, COS и TAN	Синус в радианах/Косинус в радианах/Тангенс в радианах	стр. 2-231
ASIN, ACOS и ATAN	Арксинус/Арккосинус/Арктангенс	стр. 2-234
SQRT	Квадратный корень	стр. 2-237
LN и LOG	Натуральный логарифм/Десятичный логарифм	стр. 2-240
EXP	Экспонента	стр. 2-244
EXPT (**)	Возведение в степень	стр. 2-247
Inc и Dec	Увеличение на единицу/Уменьшение на единицу	стр. 2-253
Rand	Случайное число	стр. 2-255
AryAdd	Сложение массивов	стр. 2-257
AryAddV	Прибавление значения к массиву	стр. 2-259
ArySub	Вычитание массивов	стр. 2-261
ArySubV	Вычитание значения из массива	стр. 2-263
AryMean	Среднее значение массива	стр. 2-265

Команда	Имя	Стр.
ArySD	Среднеквадратическое отклонение элементов массива	стр. 2-267
ModReal	Вещественное деление по модулю	стр. 2-269
Fraction	Дробная часть вещественного числа	стр. 2-272
CheckReal	Проверка вещественного числа	стр. 2-274

ADD (+)

Команда ADD (+) складывает целые или вещественные числа. Она также объединяет текстовые строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ADD (+)	Сложение	FUN	 	Out:=In1 + In2;

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Сложение значений	Вход	Складываемые числа Лестничная диаграмма (LD): N = 2...5 Структурированный текст (ST): N = 2*1	Зависит от типа данных.	---	0*2
Out	Выходное значение	Выход	Выходное значение	Зависит от типа данных.	---	---

*1. В одном выражении можно использовать несколько команд в качестве операторов, например: result := val1 + val2 + val3;.
Одно выражение может содержать до 64 команд.

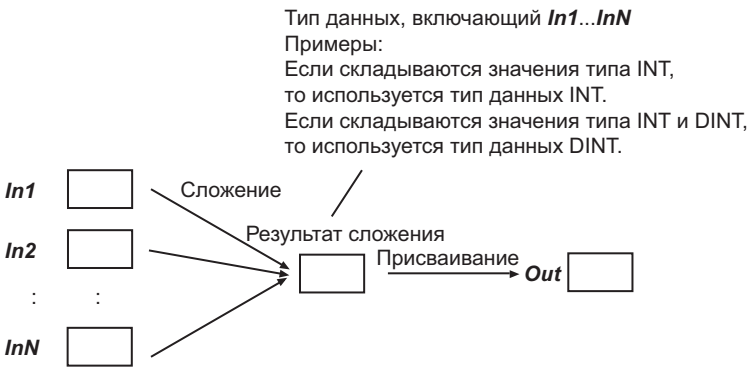
*2. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3 и будут опущены входные параметры, подключаемые к *In1* и *In2*, то для них будут применены значения по умолчанию. Но если будет опущен входной параметр, подключаемый к *In3*, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					OK
Out						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					OK

Функция

В программе на языке LD команда Add (+) складывает от двух до пяти целых или вещественных чисел и выводит результат в выходное значение *Out*.
В программе на языке ST команда Add складывает два целых или вещественных числа и выводит результат в выходное значение *Out*.

Переменные *In1*...*InN* (складываемые значения) могут относиться к разным типам данных. Но если типы данных переменных не совпадают, один из используемых типов должен поддерживать значения, допускаемые всеми остальными типами данных. Вычисления выполняются в расчете на тип данных, который включает все возможные значения остальных типов данных, используемых в команде. Например, если *In1* и *In2* являются значениями типа INT и DINT соответственно, то вычисления выполняются над значениями типа DINT. Результатом сложения в этом случае будет значение типа DINT.
О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.



Обработка при переполнении

Переполнение происходит, если сумма переменных *In1*...*InN* превышает допустимый диапазон типа данных результата сложения. В таблице ниже указаны тип данных результата сложения и значение результата сложения в случае переполнения в зависимости от типа данных слагаемых *In1*...*InN*.

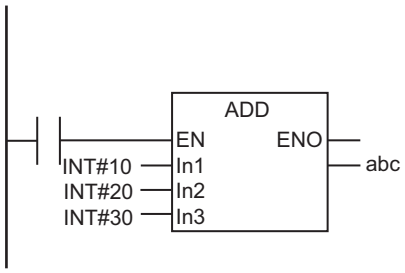
Типы данных *In1*...*InN*	Тип данных результата сложения	Значение результата сложения
Все значения являются целыми	Целочисленное значение	Результатом сложения переменных *In1*...*InN* будет значение, которое может быть выражено количеством битов, соответствующим типу данных результата сложения.*1
По крайней мере одно вещественное число	Вещественное значение	±∞*2

*1. Например, если *In1* и *In2* будут иметь значения INT#32767 и INT#3, то результат сложения будет иметь тип данных INT. Значение результата сложения будет равно содержимому младших 16 битов суммы (32 770), т. е. INT#-32766.
*2. Если сумма переменных *In1*...*InN* положительна, результатом сложения будет положительная бесконечность. Если сумма отрицательна, то результатом сложения будет отрицательная бесконечность.

Примеры способов записи

Ниже показан пример, в котором переменные *In1*, *In2* и *In3* имеют значения INT#10, INT#20 и INT#30 соответственно. Значение переменной *abc* типа INT будет равно INT#60.

LD



ST

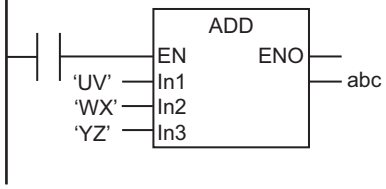
```
abc:=INT#10+INT#20+INT#30;
```

Команда ADD складывает значения от ***In1*** до ***InN.***
Вычисляется 10 + 20 + 30 = 60, поэтому ***abc*** будет равно INT#60.

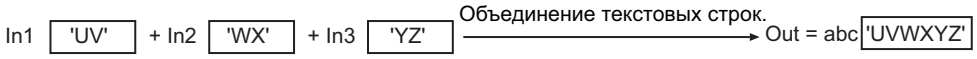
Объединение текстовых строк

Если *In1*...*InN* содержат значения типа STRING, текстовые строки соединяются в одну строку. Однако если *In1*...*InN* содержат значения типа STRING, то команду следует использовать в программе на языке LD.
В представленном ниже примере переменные *In1*, *In2* и *In3* имеют значения «UV», «WX» и «UZ» соответственно. Значение переменной *abc* типа STRING будет равно «UVWXYZ».

LD



Текстовые строки ***In1***...***InN*** объединяются в одну текстовую строку.
Вычисляется 'UV' + 'WX' + 'YZ' = 'UVWXYZ', поэтому ***abc*** будет равно 'UVWXYZ'.



Различия между языками LD и ST

Характеристики данной команды зависят от используемого языка программирования (LD или ST). В следующей таблице приводятся различающиеся характеристики. В программе на языке релейно-контактных схем (LD) характеристики команд ADD и + ничем не отличаются.

Параметр	Лестничная диаграмма (LD)	Структурированный текст (ST)
Максимальное количество складываемых значений	5	2*1
Пропуск входных параметров для складываемых значений	Можно опускать все входные параметры, кроме параметра, подключаемого к *InN*.	Никакие входные параметры опускать нельзя.
Наличие *EN* и *ENO*	Есть	Нет
Разрядность вычислений, когда все слагаемые являются целыми числами	8, 16, 32 или 64*2	32 или 64*3
Объединение текстовых строк	Возможно	Невозможно
Ошибки	Если размер строки, полученной в результате объединения текстовых строк, превышает 1986 байт, возникает ошибка.	Нет

*1. В одном выражении можно использовать несколько команд Add в качестве операторов, например: result := val1 + val2 + val3;. Одно выражение может содержать до 64 команд.

*2. Количество обрабатываемых битов определяется типом данных с самым широким диапазоном возможных значений среди используемых типов данных слагаемых. Например, если складываются значения типа SINT, INT и DINT, длина обрабатываемых значений приводится к типу данных DINT, т. е. вычисления выполняются для 32-битных значений.

*3. Если среди слагаемых нет значений типа LINT или ULINT, то выполняются 32-битные вычисления. Например, при сложении двух значений типа SINT обрабатываются 32-битные значения. Если же среди слагаемых имеются значения типа LINT или ULINT, то вычисления выполняются с 64-битными значениями.

Дополнительная информация

- При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.
- Для объединения текстовых строк в программе на языке структурированного текста (ST) используйте команду *CONCAT* на стр. 2-626.

Меры предосторожности для обеспечения надлежащей эксплуатации

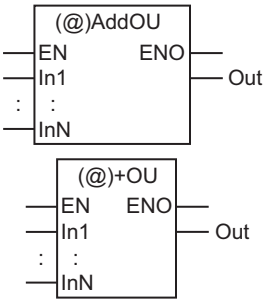
- Тип данных *Out* может отличаться от типа данных результата сложения. Он, однако, должен поддерживать весь диапазон допустимых значений типа данных результата сложения. В противном случае произойдет ошибка сборки.
  О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
- При объединении текстовых строк для *In1*...*InN* и *Out* должен использоваться тип данных STRING.
- Потеря значимости или переполнение при сложении не приводят к возникновению ошибки.
- Если при сложении происходит потеря значимости или переполнение, результат вычислений может отличаться от ожидаемого. Чтобы потери значимости или переполнения не возникало, выбирайте для входных и выходных параметров типы данных с достаточным запасом разрядности.
- В таблице ниже приведены результаты сложения для случаев, когда одним из операндов является вещественное число, а вторым — положительная или отрицательная бесконечность.

Сложение	Результат сложения
+∞ плюс число	+ ∞
-∞ плюс число	-∞
+∞ плюс +∞	+ ∞
-∞ плюс-∞	-∞
+∞ плюс-∞	Нечисловое значение

- Если какая-либо из переменных *In1*...*InN* содержит нечисловое значение, значение результата сложения является также нечисловым.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Размер объединенной текстовой строки превышает 1986 байт.

AddOU (+OU)

Команда AddOU (+OU) складывает целые и вещественные числа. Она также проверяет результат сложения целых чисел на переполнение.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AddOU (+OU)	Сложение с проверкой на переполнение	FUN		Out:=AddOU(In1, ⋯, InN);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Сложение значений	Вход	Складываемые числа N = 2...5	Зависит от типа данных.	---	0*1
Out	Выходное значение	Выход	Выходное значение	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3 и будут опущены входные параметры, подключаемые к *In1* и *In2*, то для них будут применены значения по умолчанию. Но если будет опущен входной параметр, подключаемый к *In3*, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественное число		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК *1	ОК *1					
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					

*1. Если какая-либо из переменных *In1*...*InN* содержит значение типа REAL, проверка на переполнение не производится.

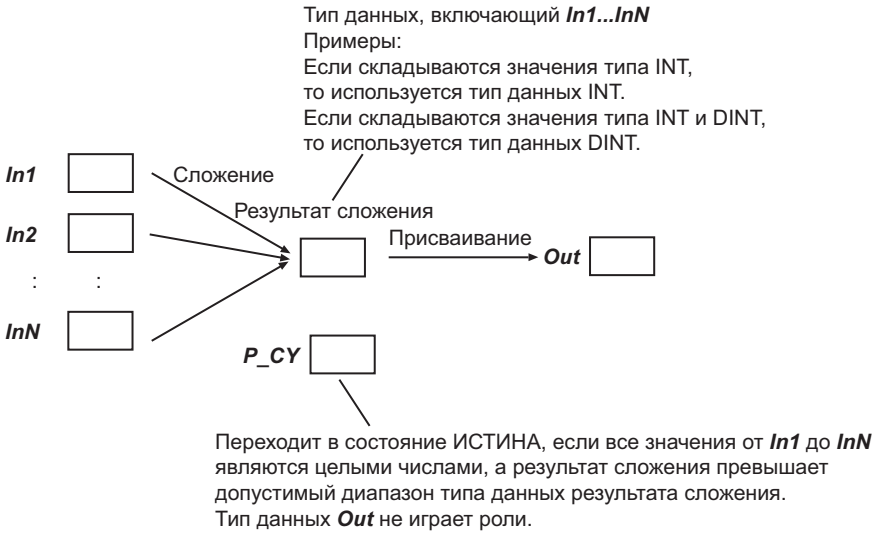
Функция

Команда AddOU (+OU) складывает от двух до пяти целых или вещественных чисел и выводит результат в выходное значение *Out*.

Переменные *In1*...*InN* (складываемые значения) могут относиться к разным типам данных. Но если типы данных переменных не совпадают, один из используемых типов должен поддерживать значения, допускаемые всеми остальными типами данных. Вычисления выполняются в расчете на тип данных, который включает все возможные значения остальных типов данных, используемых в команде. Например, если *In1* и *In2* являются значениями типа INT и DINT соответственно, то вычисления выполняются над значениями типа DINT. Результатом сложения в этом случае будет значение типа DINT.
О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Обработка при переполнении

Переполнение происходит, если сумма переменных *In1*...*InN* превышает допустимый диапазон типа данных результата сложения. Если все переменные *In1*...*InN* содержат целые значения, то при возникновении переполнения значение системной переменной *P_CY* (Флаг переноса) меняется на ИСТИНА.
Если какая-либо из переменных *In1*...*InN* содержит значение типа REAL, проверка на переполнение не производится. Поэтому значение *P_CY* не изменится.



В таблице ниже указаны тип данных результата сложения, значение результата сложения и значение флага *P_CY* в случае переполнения в зависимости от типа данных слагаемых *In1*...*InN*.

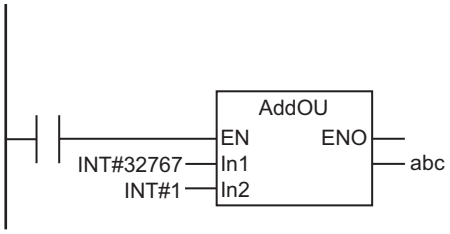
Типы данных *In1*...*InN*	Тип данных результата сложения	Значение результата сложения	Значение *P_CY*
Все значения являются целыми	Целочисленное значение	Результатом сложения переменных *In1*...*InN* будет значение, которое может быть выражено количеством битов, соответствующим типу данных результата сложения.*1	ИСТИНА
По крайней мере одно вещественное число	Вещественное значение	±∞*2	Не меняется.

*1. Например, если *In1* и *In2* будут иметь значения INT#32767 и INT#3, то результат сложения будет иметь тип данных INT. Значение результата сложения будет равно содержимому младших 16 битов суммы (32 770), т. е. INT#-32766.
*2. Если сумма переменных *In1*...*InN* положительна, результатом сложения будет положительная бесконечность. Если сумма отрицательна, то результатом сложения будет отрицательная бесконечность.

Примеры способов записи

В приведенном ниже примере переменные *In1* и *In2* содержат значения INT#32767 и INT#1, а переменная *abc* относится к типу данных INT.
Поскольку обе переменные, *In1* и *In2*, содержат значения типа INT, то и результатом их сложения будет значение типа INT. Сумма двух значений (32 768) выходит за допустимый диапазон типа данных INT, поэтому значение *P_CY* меняется на ИСТИНА. Переменная *abc* типа INT примет значение INT#-32768 (младшие 16 битов числа 32768).

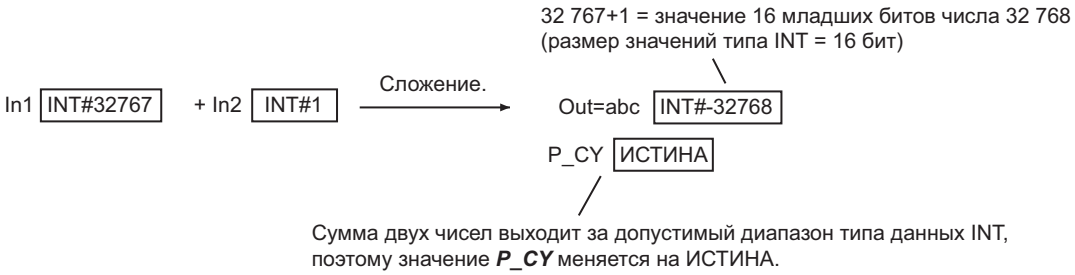
LD



ST

```
abc:=AddOU(INT#32767, INT#1);
```

Команда AddOU добавляет ***In1*** к ***InN***.
Сумма двух чисел (32 768) превышает допустимый диапазон типа данных INT, поэтому значение ***P_CY*** меняется на ИСТИНА.



Различия между языками LD и ST

Характеристики данной команды не зависят от используемого языка программирования (LD или ST). В лестничной диаграмме (LD) команда AddOU и команда +OU имеют одинаковые характеристики.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
P_CY	Флаг переноса (CY)	BOOL	ИСТИНА: при целочисленных вычислениях произошло переполнение. ЛОЖЬ: при целочисленных вычислениях не произошло переполнения.

Дополнительная информация

- Если переменная *Out* имеет тип данных REAL, используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не содержит ли она положительную или отрицательную бесконечность либо нечисловое значение.

- Если выполнять проверку на переполнение не требуется, используйте команду *ADD (+)* на стр. 2-195. Это сократит время обработки.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Тип данных *Out* может отличаться от типа данных результата сложения. Он, однако, должен поддерживать весь диапазон допустимых значений типа данных результата сложения. В противном случае произойдет ошибка сборки.
  О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
- Если при сложении происходит потеря значимости или переполнение, результат вычислений может отличаться от ожидаемого. Чтобы потери значимости или переполнения не возникало, выбирайте для входных и выходных параметров типы данных с достаточным запасом разрядности.
- В таблице ниже приведены результаты сложения для случаев, когда одним из операндов является вещественное число, а вторым — положительная или отрицательная бесконечность.

Сложение	Результат сложения
+∞ плюс число	+ ∞
-∞ плюс число	-∞
+∞ плюс +∞	+ ∞
-∞ плюс-∞	-∞
+∞ плюс-∞	Нечисловое значение

- Если какая-либо из переменных *In1*...*InN* содержит нечисловое значение, значение результата сложения является также нечисловым.

SUB (-)

Команда SUB (-) вычитает целые и вещественные числа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SUB (-)	Вычитание	FUN	(@)SUB EN ENO In1 Out In2 (@)- EN ENO In1 Out In2	Out:=In1 - In2;

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Уменьшаемое	Вход	Уменьшаемое	Зависит от типа данных.	---	0*1
In2	Вычитаемое		Вычитаемое			
Out	Выходное значение	Выход	Выходное значение	Зависит от типа данных.	---	---

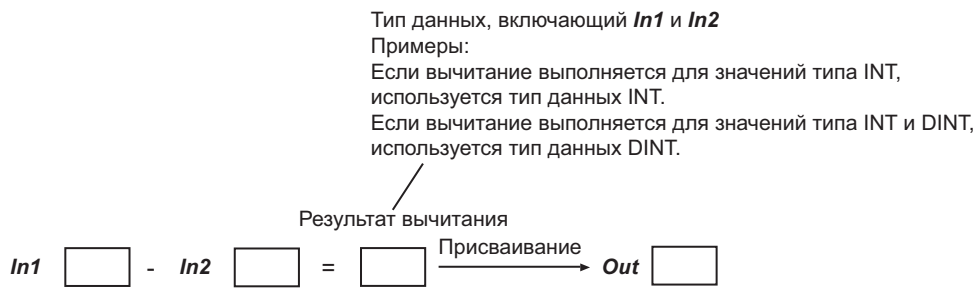
*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
In2						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					

Функция

Команда SUB (-) вычитает вычитаемое *In2* из уменьшаемого *In1* и выводит результат в выходное значение *Out*.

Типы данных параметров *In1* и *In2* могут отличаться. Но если типы данных параметров не совпадают, один из используемых типов должен поддерживать значения, допускаемые другим типом данных. Вычисления выполняются в расчете на тип данных, который включает все возможные значения второго типа данных, используемого в команде. Например, если *In1* и *In2* являются значениями типа INT и DINT соответственно, то вычисления выполняются над значениями типа DINT. Результатом вычитания в этом случае будет значение типа DINT.



О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Обработка при переполнении

Переполнение происходит, если разность значений *In1* и *In2* превышает допустимый диапазон типа данных результата вычитания. В таблице ниже указаны тип данных результата вычитания и значение результата вычитания в случае переполнения в зависимости от типа данных переменных *In1* и *In2*.

Типы данных *In1* и *In2*	Тип данных результата вычитания	Значение результата вычитания
Все значения являются целыми	Целочисленное значение	Результатом вычитания переменных *In1* и *In2* будет значение, которое может быть выражено количеством битов, соответствующим типу данных результата вычитания.*1
По крайней мере одно вещественное число	Вещественное значение	±∞*2

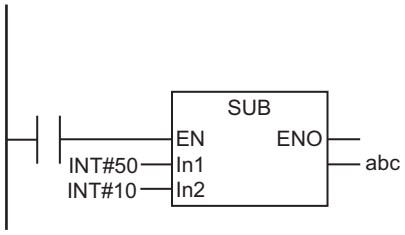
*1. Например, если *In1* и *In2* будут иметь значения INT#32767 и INT#-3, то результат вычитания будет иметь тип данных INT. Значение результата вычитания будет равно содержимому младших 16 битов разницы (32 770), т. е. INT#-32766.

*2. Если разница значений *In1* и *In2* положительна, результатом вычитания будет положительная бесконечность. Если разница отрицательна, то результатом вычитания будет отрицательная бесконечность.

Примеры способов записи

Ниже показан пример, в котором переменные *In1* и *In2* имеют значения INT#50 и INT#10 соответственно. Значение переменной *abc* типа INT будет равно INT#40.

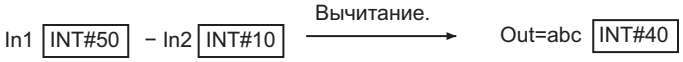
LD



ST

```
abc:=INT#50-INT#10;
```

Команда SUB вычитает ***In2*** из ***In1***.
Вычисляется 50 − 10 = 40, поэтому ***abc*** будет равно INT#40.



Различия между языками LD и ST

Характеристики данной команды зависят от используемого языка программирования (LD или ST). В следующей таблице приводятся различающиеся характеристики. В программе на языке релейно-контактных схем (LD) характеристики команд SUB и - ничем не отличаются.

Параметр	Лестничная диаграмма (LD)	Структурированный текст (ST)
Наличие *EN* и *ENO*	Есть	Нет
Разрядность вычислений, когда уменьшаемое и вычитаемое являются целыми числами	8, 16, 32 или 64*1	32 или 64*2

*1. Количество обрабатываемых битов определяется типом данных с самым широким диапазоном возможных значений среди используемых типов данных уменьшаемого и вычитаемого. Например, если операндами являются значения типа SINT и DINT, длина обрабатываемых значений приводится к типу данных DINT, т. е. вычисления выполняются для 32-битных значений.

*2. Если ни уменьшаемое, ни вычитаемое не содержат значений типа LINT или ULINT, то выполняются 32-битные вычисления. Например, если одно значение типа SINT вычитается из другого значения типа SINT, обрабатываются 32-битные значения. Если же уменьшаемое или вычитаемое содержит значение типа LINT или ULINT, то выполняются 64-битные вычисления.

Дополнительная информация

При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Тип данных *Out* может отличаться от типа данных результата вычитания. Он, однако, должен поддерживать весь диапазон допустимых значений типа данных результата вычитания. В противном случае произойдет ошибка сборки.
  О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
- Потеря значимости или переполнение при вычитании не приводят к возникновению ошибки.
- Если при вычитании происходит потеря значимости или переполнение, результат вычислений может отличаться от ожидаемого. Чтобы потери значимости или переполнения не возникало, выбирайте для входных и выходных параметров типы данных с достаточным запасом разрядности.
- В таблице ниже приведены результаты вычитания для случаев, когда одним из операндов является вещественное число, а вторым — положительная или отрицательная бесконечность.

Вычитание	Результат вычитания
+∞ минус число	+ ∞
Число минус +∞	-∞

Вычитание	Результат вычитания
-∞ минус число	-∞
Число минус-∞	+ ∞
+∞ минус +∞	Нечисловое значение
+∞ минус -∞	+ ∞
-∞ минус +∞	-∞
-∞ минус -∞	Нечисловое значение

- Если какой-либо из входов, *In1* или *In2*, содержит нечисловое значение, результатом вычитания также будет нечисловое значение.

SubOU (-OU)

Команда SubOU (-OU) вычитает целые или вещественные числа. Она также проверяет результат вычитания целых чисел на переполнение.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SubOU (-OU)	Вычитание с проверкой на переполнение	FUN	(@)SubOU EN ENO In1 Out In2 (@)-OU EN ENO In1 Out In2	Out:=SubOU(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Уменьшаемое	Вход	Уменьшаемое	Зависит от типа данных.	---	0*1
In2	Вычитаемое		Вычитаемое			
Out	Выходное значение	Выход	Выходное значение	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	OK *1	OK *1					
In2						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	OK *1	OK *1					
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					

*1. Если какая-либо из переменных, *In1* или *In2*, содержит значение типа REAL, проверка на переполнение не производится.

Функция

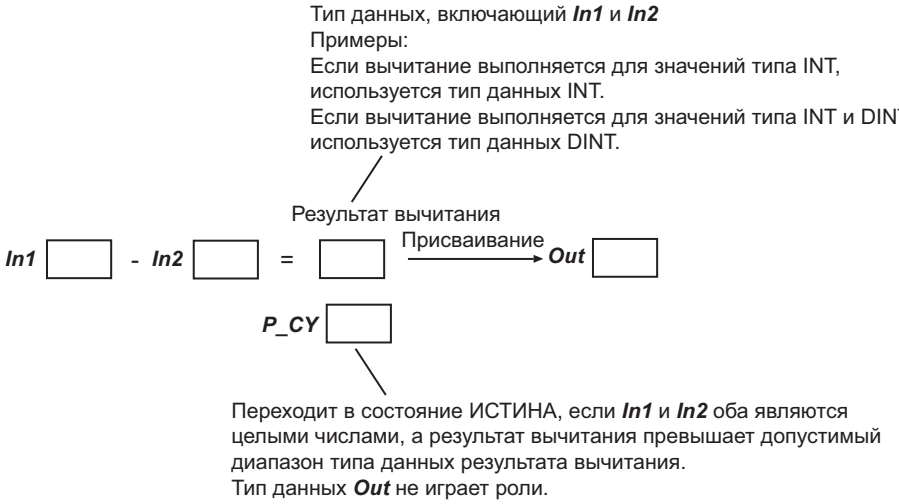
Команда SubOU (-OU) вычитает вычитаемое *In2* из уменьшаемого *In1* и выводит результат в выходное значение *Out*.

Типы данных параметров *In1* и *In2* могут отличаться. Но если типы данных параметров не совпадают, один из используемых типов должен поддерживать значения, допускаемые другим

типом данных. Вычисления выполняются в расчете на тип данных, который включает все возможные значения второго типа данных, используемого в команде. Например, если *In1* и *In2* являются значениями типа INT и DINT соответственно, то вычисления выполняются над значениями типа DINT. Результатом вычитания в этом случае будет значение типа DINT.
О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Обработка при переполнении

Переполнение происходит, если разность значений *In1* и *In2* превышает допустимый диапазон типа данных результата вычитания. Если переменные *In1* и *In2* содержат целые значения, то при возникновении переполнения значение системной переменной *P_CY* (Флаг переноса) меняется на ИСТИНА.
Если какая-либо из переменных, *In1* или *In2*, содержит значение типа REAL, проверка на переполнение не производится. Поэтому значение *P_CY* не изменится.



В таблице ниже указаны тип данных результата вычитания, значение результата вычитания и значение флага *P_CY* в случае переполнения в зависимости от типа данных переменных *In1* и *In2*.

Типы данных *In1* и *In2*	Тип данных результата вычитания	Значение результата вычитания	Значение *P_CY*
Все значения являются целыми	Целочисленное значение	Результатом вычитания переменных *In1* и *In2* будет значение, которое может быть выражено количеством битов, соответствующим типу данных результата вычитания.*1	ИСТИНА
По крайней мере одно вещественное число	Вещественное значение	±∞*2	Не меняется.

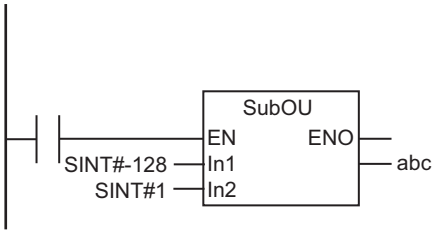
*1. Например, если *In1* и *In2* будут иметь значения INT#32767 и INT#-3, то результат вычитания будет иметь тип данных INT. Значение результата вычитания будет равно содержимому младших 16 битов разницы (32 770), т. е. INT#-32766.

*2. Если разница значений *In1* и *In2* положительна, результатом вычитания будет положительная бесконечность. Если разница отрицательна, то результатом вычитания будет отрицательная бесконечность.

Примеры способов записи

В приведенном ниже примере переменные *In1* и *In2* содержат значения SINT#-128 и INT#1, а переменная *abc* относится к типу данных SINT.
Поскольку обе переменные, *In1* и *In2*, содержат значения типа SINT, то и результатом вычитания будет значение типа SINT. Разница двух этих значений (-129) выходит за допустимый диапазон типа данных SINT, поэтому значение *P_CY* меняется на ИСТИНА. Переменная *abc* типа SINT примет значение SINT#127 (значение младших восьми битов числа -129).

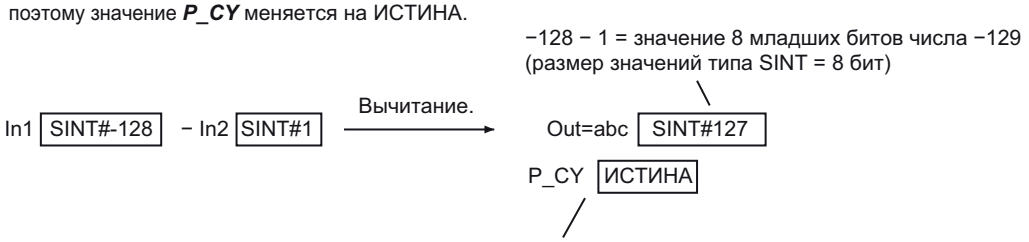
LD



ST

```
abc:=SubOU(SINT#-128, SINT#1);
```

Команда SubOU вычитает ***In2*** из ***In1***.
Разница этих двух значений (-129) превышает допустимый диапазон типа данных SINT, поэтому значение ***P_CY*** меняется на ИСТИНА.



Различия между языками LD и ST

Характеристики данной команды не зависят от используемого языка программирования (LD или ST). В лестничной диаграмме (LD) команда SubOU и команда -OU имеют одинаковые характеристики.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
P_CY	Флаг переноса (CY)	BOOL	ИСТИНА: при целочисленных вычислениях произошло переполнение. ЛОЖЬ: при целочисленных вычислениях не произошло переполнения.

Дополнительная информация

- При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

- Если выполнять проверку на переполнение не требуется, используйте команду *SUB (-)* на стр. 2-204. Это сократит время обработки.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Тип данных *Out* может отличаться от типа данных результата вычитания. Он, однако, должен поддерживать весь диапазон допустимых значений типа данных результата вычитания. В противном случае произойдет ошибка сборки.
  О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
- Если при вычитании происходит потеря значимости или переполнение, результат вычислений может отличаться от ожидаемого. Чтобы потери значимости или переполнения не возникало, выбирайте для входных и выходных параметров типы данных с достаточным запасом разрядности.
- В таблице ниже приведены результаты вычитания для случаев, когда одним из операндов является вещественное число, а вторым — положительная или отрицательная бесконечность.

Вычитание	Результат вычитания
+∞ минус число	+ ∞
Число минус +∞	-∞
-∞ минус число	-∞
Число минус-∞	+ ∞
+∞ минус +∞	Нечисловое значение
+∞ минус -∞	+ ∞
-∞ минус +∞	-∞
-∞ минус -∞	Нечисловое значение

- Если какой-либо из входов, *In1* или *In2*, содержит нечисловое значение, результатом вычитания также будет нечисловое значение.

MUL (*)

Команда MUL(*) умножает целые и вещественные числа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MUL (*)	Умножение	FUN	(@)MUL EN ENO In1 Out : : InN (@)* EN ENO In1 Out : : InN	Out:=In1 * In2;

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Перемножаемые значения	Вход	Перемножаемые числа Лестничная диаграмма (LD): N = 2...5 Структурированный текст (ST): N = 2*1	Зависит от типа данных.	---	1*2
Out	Выходное значение	Выход	Выходное значение	Зависит от типа данных.	---	---

*1. В одном выражении можно использовать несколько команд MUL в качестве операторов, например: result := val1 * val2 * val3;.
Одно выражение может содержать до 64 команд.

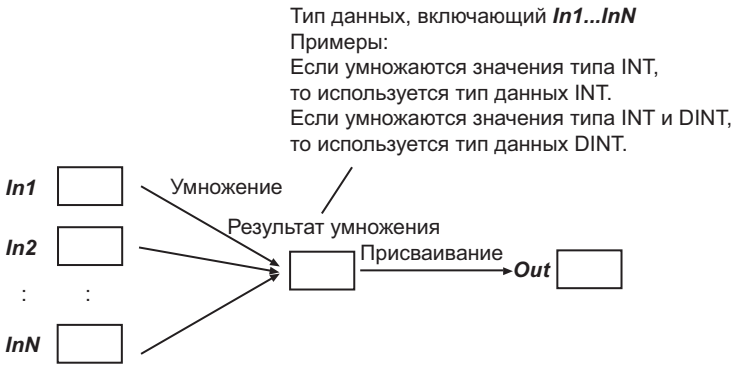
*2. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3 и будут опущены входные параметры, подключаемые к *In1* и *In2*, то для них будут применены значения по умолчанию. Но если будет опущен входной параметр, подключаемый к *In3*, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					

Функция

В программе на языке LD команда MUL (*) перемножает от двух до пяти целых или вещественных чисел и выводит результат в выходное значение *Out*.
В программе на языке ST команда MUL перемножает два целых или вещественных числа и выводит результат в выходное значение *Out*.

Переменные *In1*...*InN* (перемножаемые значения) могут относиться к разным типам данных. Но если типы данных переменных не совпадают, один из используемых типов должен поддерживать значения, допускаемые всеми остальными типами данных. Вычисления выполняются в расчете на тип данных, который включает все возможные значения остальных типов данных, используемых в команде. Например, если *In1* и *In2* являются значениями типа INT и DINT соответственно, то вычисления выполняются над значениями типа DINT. Результатом умножения в этом случае будет значение типа DINT.
О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.



Обработка при переполнении

Переполнение происходит, если произведение переменных *In1*...*InN* превышает допустимый диапазон типа данных результата умножения. В таблице ниже указаны тип данных результата умножения и значение результата умножения в случае переполнения в зависимости от типа данных множителей *In1*...*InN*.

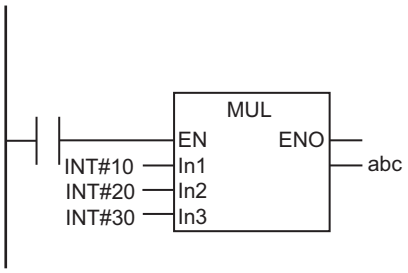
Типы данных *In1*...*InN*	Тип данных результата умножения	Значение результата умножения
Все значения являются целыми	Целочисленное значение	Результатом умножения переменных *In1*...*InN* будет значение, которое может быть выражено количеством битов, соответствующим типу данных результата умножения.*1
По крайней мере одно вещественное число	Вещественное значение	±∞*2

*1. Например, если *In1* и *In2* будут иметь значения INT#16384 и INT#2, то результат умножения будет иметь тип данных INT. Значение результата умножения будет равно содержимому младших 16 битов произведения (32 768), т. е., INT#-32768.
*2. Если произведение переменных *In1*...*InN* положительно, результатом умножения будет положительная бесконечность. Если произведение отрицательно, то результатом умножения будет отрицательная бесконечность.

Примеры способов записи

Ниже показан пример, в котором переменные *In1*, *In2* и *In3* имеют значения INT#10, INT#20 и INT#30 соответственно. Значение переменной *abc* типа INT будет равно INT#6000.

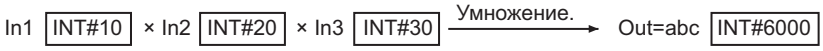
LD



ST

```
abc:=INT#10*INT#20*INT#30;
```

Команда MUL перемножает значения ***In1...InN***.
Вычисляется 10 × 20 × 30 = 6000, поэтому ***abc*** будет равно INT#6000.



Различия между языками LD и ST

Характеристики данной команды зависят от используемого языка программирования (LD или ST). В следующей таблице приводятся различающиеся характеристики. В программе на языке релейно-контактных схем (LD) характеристики команд MUL и * ничем не отличаются.

Параметр	Лестничная диаграмма (LD)	Структурированный текст (ST)
Максимальное количество умножаемых значений	5	2*1
Пропуск входных параметров для умножаемых значений	Можно опускать все входные параметры, кроме параметра, подключаемого к InN.	Никакие входные параметры опускать нельзя.
Наличие *EN* и *ENO*	Есть	Нет
Разрядность вычислений, когда все множители являются целыми числами	8, 16, 32 или 64*2	32 или 64*3

*1. В одном выражении можно использовать несколько команд MUL в качестве операторов, например: result := val1 * val2 * val3;. Одно выражение может содержать до 64 команд.

*2. Количество обрабатываемых битов определяется типом данных с самым широким диапазоном возможных значений среди используемых типов данных множителей. Например, если перемножаются значения типа SINT, INT и DINT, длина обрабатываемых значений приводится к типу данных DINT, т. е. вычисления выполняются для 32-битных значений.

*3. Если среди множителей нет значений типа LINT или ULINT, то выполняются 32-битные вычисления. Например, при умножении двух значений типа SINT обрабатываются 32-битные значения. Если же среди множителей имеются значения типа LINT или ULINT, то вычисления выполняются с 64-битными значениями.

Дополнительная информация

При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

Меры предосторожности для обеспечения надлежащей эксплуатации

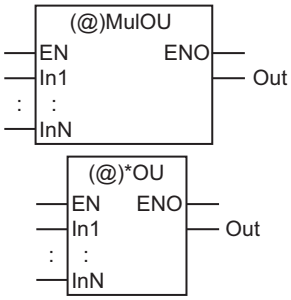
- Тип данных *Out* может отличаться от типа данных результата умножения. Он, однако, должен поддерживать весь диапазон допустимых значений типа данных результата умножения. В противном случае произойдет ошибка сборки.
  О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
- Потеря значимости или переполнение при умножении не приводят к возникновению ошибки.
- Если при умножении происходит потеря значимости или переполнение, результат вычислений может отличаться от ожидаемого. Чтобы потери значимости или переполнения не возникало, выбирайте для входных и выходных параметров типы данных с достаточным запасом разрядности.
- В таблице ниже приведены результаты умножения для случаев, когда одним из операндов является вещественное число, а вторым — положительная или отрицательная бесконечность.

Умножение	Результат умножения
+ ∞ умножить на положительное число	+ ∞
+ ∞ умножить на отрицательное число	-∞
- ∞ умножить на положительное число	-∞
- ∞ умножить на отрицательное число	+ ∞
+ ∞ умножить на + ∞	+ ∞
- ∞ умножить на - ∞	+ ∞
+ ∞ умножить на - ∞	-∞
+ ∞ умножить на 0	Нечисловое значение
-∞ умножить на 0	Нечисловое значение

- Если какая-либо из переменных *In1*...*InN* содержит нечисловое значение, значение результата умножения является также нечисловым.

MulOU (*OU)

Команда MulOU (*OU) умножает целые и вещественные числа и выводит результат. Она также проверяет результат умножения целых чисел на переполнение.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MulOU (*OU)	Умножение с проверкой на переполнение	FUN		Out:=MulOU(In1, ⋯, InN);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Перемножаемые значения	Вход	Перемножаемые числа N = 2...5	Зависит от типа данных.	---	1*1
Out	Выходное значение	Выход	Выходное значение	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3 и будут опущены входные параметры, подключаемые к *In1* и *In2*, то для них будут применены значения по умолчанию. Но если будет опущен входной параметр, подключаемый к *In3*, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN						OK	OK	OK	OK	OK	OK	OK	OK	OK *1	OK *1					
Out						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					

*1. Если какая-либо из переменных *In1*...*InN* содержит значение типа REAL, проверка на переполнение не производится.

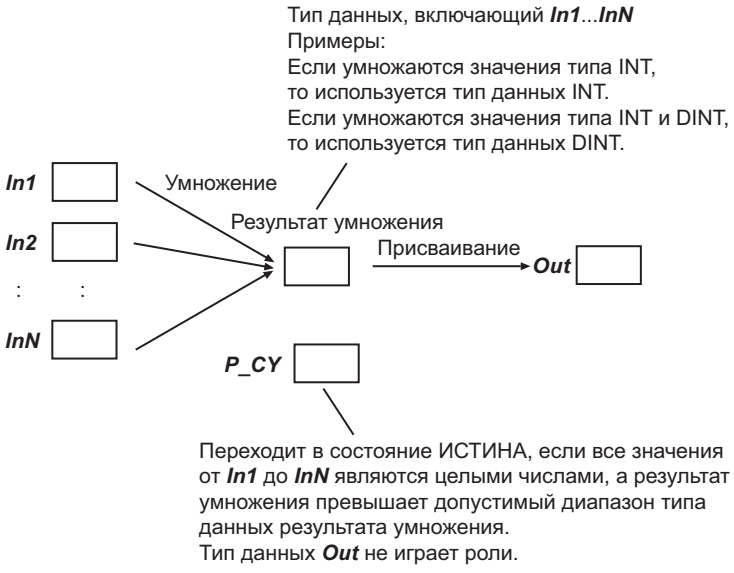
Функция

Команда MulOU (*OU) перемножает от двух до пяти целых или вещественных чисел и выводит результат в выходное значение *Out*.

Перемножаемые значения *In1*...*InN* могут относиться к разным типам данных. Но если типы данных переменных не совпадают, один из используемых типов должен поддерживать значения, допускаемые всеми остальными типами данных. Вычисления выполняются в расчете на тип данных, который включает все возможные значения остальных типов данных, используемых в команде. Например, если *In1* и *In2* являются значениями типа INT и DINT соответственно, то вычисления выполняются над значениями типа DINT. Результатом умножения в этом случае будет значение типа DINT.
О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Обработка при переполнении

Переполнение происходит, если произведение переменных *In1*...*InN* превышает допустимый диапазон типа данных результата умножения. Если все переменные *In1*...*InN* содержат целые значения, то при возникновении переполнения значение системной переменной *P_CY* (Флаг переноса) меняется на ИСТИНА.
Если какая-либо из переменных *In1*...*InN* содержит значение типа REAL, проверка на переполнение не производится. Поэтому значение *P_CY* не изменится.



В таблице ниже указаны тип данных результата умножения, значение результата умножения и значение флага *P_CY* в случае переполнения в зависимости от типа данных множителей *In1*...*InN*.

Типы данных *In1*...*InN*	Тип данных результата умножения	Значение результата умножения	Значение *P_CY*
Все значения являются целыми	Целочисленное значение	Результатом умножения переменных *In1*...*InN* будет значение, которое может быть выражено количеством битов, соответствующим типу данных результата умножения.*1	ИСТИНА

Типы данных *In1...InN*	Тип данных результата умножения	Значение результата умножения	Значение *P_CY*
По крайней мере одно вещественное число	Вещественное значение	$\pm\infty$*2	Не меняется.

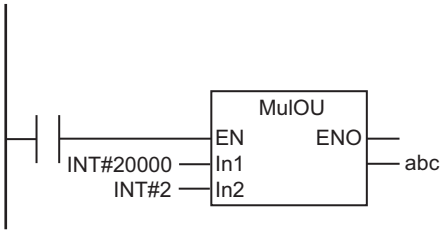
*1. Например, если *In1* и *In2* будут иметь значения INT#16384 и INT#2, то результат умножения будет иметь тип данных INT. Значение результата умножения будет равно содержимому младших 16 битов произведения (32 768), т. е., INT#-32768.

*2. Если произведение переменных *In1...InN* положительно, результатом умножения будет положительная бесконечность. Если произведение отрицательно, то результатом умножения будет отрицательная бесконечность.

Примеры способов записи

В приведенном ниже примере переменные *In1* и *In2* содержат значения INT#20000 и INT#2, а переменная *abc* относится к типу данных INT.
Поскольку обе переменные, *In1* и *In2*, содержат значения типа INT, то и результатом их умножения будет значение типа INT. Произведение двух значений (40 000) выходит за допустимый диапазон типа данных INT, поэтому значение *P_CY* меняется на ИСТИНА. Переменная *abc* типа INT примет значение INT#-25536 (младшие 16 битов числа 40000).

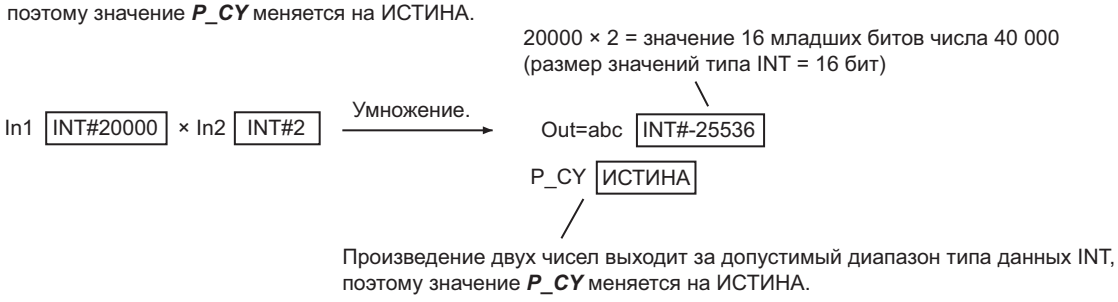
LD



ST

```
abc:=MulOU(INT#20000, INT#2);
```

Команда MulOU умножает ***In1*** на ***InN***.
Произведение двух значений (40 000) превышает допустимый диапазон типа данных INT, поэтому значение ***P_CY*** меняется на ИСТИНА.



Различия между языками LD и ST

Характеристики данной команды не зависят от используемого языка программирования (LD или ST). В лестничной диаграмме (LD) команда MulOU и команда *OU имеют одинаковые характеристики.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
P_CY	Флаг переноса (CY)	BOOL	ИСТИНА: при целочисленных вычислениях произошло переполнение. ЛОЖЬ: при целочисленных вычислениях не произошло переполнения.

Дополнительная информация

- При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.
- Если выполнять проверку на переполнение не требуется, используйте команду *MUL (*)* на стр. 2-212. Это сократит время обработки.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Тип данных *Out* может отличаться от типа данных результата умножения. Он, однако, должен поддерживать весь диапазон допустимых значений типа данных результата умножения. В противном случае произойдет ошибка сборки.
  О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
- Если при умножении происходит потеря значимости или переполнение, результат вычислений может отличаться от ожидаемого. Чтобы потери значимости или переполнения не возникало, выбирайте для входных и выходных параметров типы данных с достаточным запасом разрядности.
- В таблице ниже приведены результаты умножения для случаев, когда одним из операндов является вещественное число, а вторым — положительная или отрицательная бесконечность.

Умножение	Результат умножения
+ ∞ умножить на положительное число	+ ∞
+ ∞ умножить на отрицательное число	-∞
- ∞ умножить на положительное число	-∞
- ∞ умножить на отрицательное число	+ ∞
+ ∞ умножить на + ∞	+ ∞
- ∞ умножить на - ∞	+ ∞
+ ∞ умножить на - ∞	-∞
+ ∞ умножить на 0	Нечисловое значение
-∞ умножить на 0	Нечисловое значение

- Если какая-либо из переменных *In1*...*InN* содержит нечисловое значение, значение результата умножения является также нечисловым.

DIV (/)

Команда DIV (/) делит целые или вещественные числа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DIV (/)	Деление	FUN	(@)DIV EN ENO In1 Out In2 (@)/ EN ENO In1 Out In2	Out:=In1/ In2;

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Делимое	Вход	Делимое	Зависит от типа данных.	---	*1
In2	Делитель		Делитель			
Out	Выходное значение	Выход	Выходное значение	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

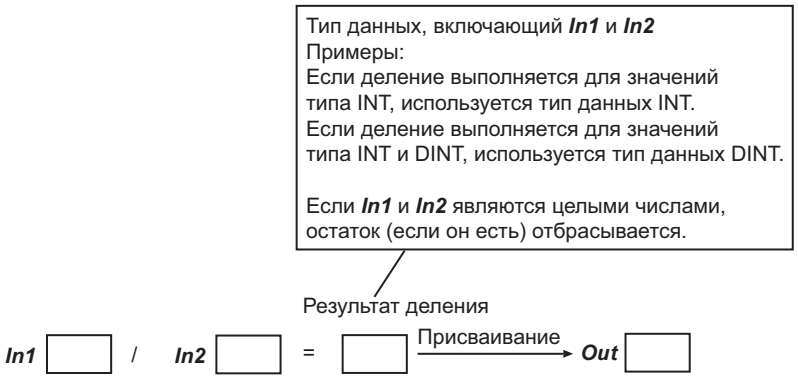
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
In2						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
Out						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					

Функция

Команда DIV (/) делит делимое *In1* на делитель *In2* и выводит результат в выходное значение *Out*.

Типы данных параметров *In1* и *In2* могут отличаться. Но если типы данных параметров не совпадают, один из используемых типов должен поддерживать значения, допускаемые другим типом данных. Вычисления выполняются в расчете на тип данных, который включает все возможные значения второго типа данных, используемого в команде. Например, если *In1* и *In2* являются значениями типа INT и DINT соответственно, то вычисления выполняются над значениями типа DINT. Результатом деления в этом случае будет значение типа DINT.
Если *In1* и *In2* являются целыми числами, то остаток (если он имеется) отбрасывается.

О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.



Обработка при переполнении

Переполнение происходит, если частное значений *In1* и *In2* превышает допустимый диапазон типа данных результата деления. В таблице ниже указаны тип данных результата деления и значение результата деления в случае переполнения в зависимости от типа данных переменных *In1* и *In2*.

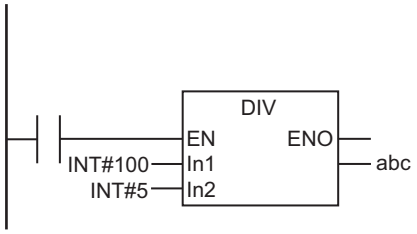
Типы данных *In1* и *In2*	Тип данных результата деления	Значение результата деления
По крайней мере одно вещественное число	Вещественное значение	±∞*1

*1. Если частное *In1* и *In2* положительно, результатом деления будет положительная бесконечность. Если частное отрицательно, то результатом деления будет отрицательная бесконечность.

Примеры способов записи

Ниже показан пример, в котором переменные *In1* и *In2* имеют значения INT#100 и INT#5 соответственно. Значение переменной *abc* типа INT будет равно INT#20.

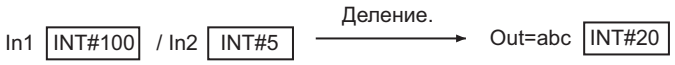
LD



ST

```
abc:=INT#100/INT#5;
```

Команда DIV делит ***In1*** на ***In2***.
Вычисляется 100/5 = 20, поэтому ***abc*** будет равно INT#20.



Различия между языками LD и ST

Характеристики данной команды зависят от используемого языка программирования (LD или ST). В следующей таблице приводятся различающиеся характеристики. В лестничных диаграммах характеристики команд DIV и / ничем не отличаются.

Параметр	Лестничная диаграмма (LD)	Структурированный текст (ST)
Наличие *EN* и *ENO*	Есть	Нет
Разрядность вычислений, когда делимое и делитель являются целыми числами	8, 16, 32 или 64*1	32 или 64*2

*1. Количество обрабатываемых битов определяется типом данных с самым широким диапазоном возможных значений среди используемых типов данных делимого и делителя. Например, если операндами являются значения типа SINT и DINT, длина обрабатываемых значений приводится к типу данных DINT, т. е. вычисления выполняются для 32-битных значений.

*2. Если ни делимое, ни делитель не содержат значений типа LINT или ULINT, то выполняются 32-битные вычисления. Например, если одно значение типа SINT делится на другое значение типа SINT, обрабатываются 32-битные значения. Если же делимое или делитель содержит значение типа LINT или ULINT, то выполняются 64-битные вычисления.

Дополнительная информация

При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

Меры предосторожности для обеспечения надлежащей эксплуатации

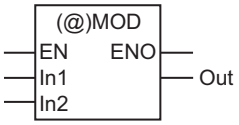
- Тип данных *Out* может отличаться от типа данных результата деления. Он, однако, должен поддерживать весь диапазон допустимых значений типа данных результата деления. В противном случае произойдет ошибка сборки.
  О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
- Потеря значимости или переполнение при делении не приводят к возникновению ошибки.
- Если при делении происходит потеря значимости или переполнение, результат вычислений может отличаться от ожидаемого. Чтобы потери значимости или переполнения не возникало, выбирайте для входных и выходных параметров типы данных с достаточным запасом разрядности.
- В таблице ниже приведены результаты деления для случаев, когда одним из операндов является вещественное число, а вторым — положительная бесконечность, отрицательная бесконечность или 0.

		In1				
		+ ∞	**Положительное число**	**0**	**Отрицательное число**	**-∞**
In2	**+ ∞**	Нечисловое значение	0	0	0	Нечисловое значение
	Положительное число	+ ∞	Положительное число	0	Отрицательное число	-∞
	0	+ ∞	+ ∞	Нечисловое значение	-∞	-∞
	Отрицательное число	-∞	Отрицательное число	0	Положительное число	+ ∞
	-∞	Нечисловое значение	0	0	0	Нечисловое значение

- Если какой-либо из входов, *In1* или *In2*, содержит нечисловое значение, результатом деления также будет нечисловое значение.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) *In1* и *In2* являются целыми числами, при этом *In2* = 0.

MOD

Команда MOD находит остаток от деления целых чисел.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MOD	Деление по модулю	FUN		Out:=In1 MOD In2;

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Делимое	Вход	Делимое	Зависит от типа данных.	---	*1
In2	Делитель		Делитель			
Out	Остаток	Выход	Остаток	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК							
In2						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК							
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК							

Функция

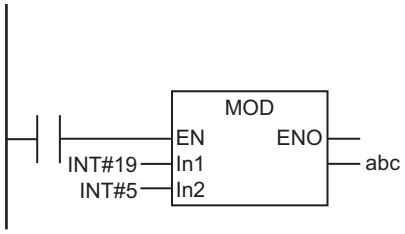
Команда MOD делит делимое *In1* на делитель *In2* и возвращает остаток от деления.
Типы данных переменных *In1*, *In2* и *Out* могут отличаться. Но если типы данных переменных не совпадают, один из используемых типов должен поддерживать значения, допускаемые всеми остальными типами данных.
О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Эта команда выполняет вычисление по следующей формуле:
Out = *In1* - (*In1*/*In2*)**In2*
Дробные части при выполнении деления отбрасываются.
В таблице ниже приведены значения *Out* для разных значений *In1* и *In2*.

Значение *In1*	Значение *In2*	Значение *Out*
5	3	2
5	-3	2
-5	3	-2
-5	-3	-2

Ниже показан пример, в котором переменные *In1* и *In2* имеют значения INT#19 и INT#5 соответственно. Значение переменной *abc* будет равно INT#4.

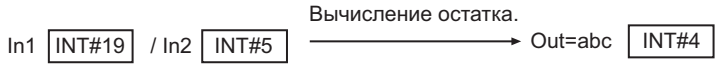
LD



ST

```
abc:=INT#19 MOD INT#5;
```

Команда MOD определяет остаток от деления ***In1*** на ***In2***.
Остаток от 19/5 равен 4, поэтому ***abc*** будет равно INT#4.



Меры предосторожности для обеспечения надлежащей эксплуатации

Выбирайте для *Out* тип данных, который поддерживает диапазоны допустимых значений переменных *In1* и *In2*.

О преобразовании разных типов данных см. в разделах *Таблица рангов типов данных* и *Правила приведения типов* в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

ABS

Команда ABS находит абсолютное значение целого или вещественного числа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ABS	Абсолютное значение	FUN	(@)ABS EN ENO In Out	Out:=ABS(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Число для обработки	Вход	Число для обработки	Зависит от типа данных.	---	*1
Out	Абсолютное значение	Выход	Абсолютное значение	Зависит от типа данных.*2	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

*2. Отрицательные числа исключены.

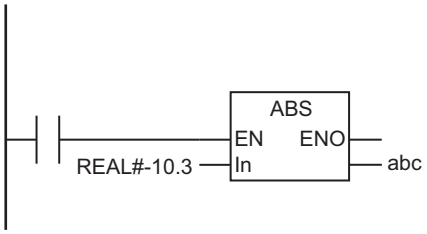
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					

Функция

Команда ABS выводит абсолютное значение числа для обработки *In*.
Тип данных *In* может отличаться от типа данных абсолютного значения *Out*.

В показанном ниже примере программы *In* = REAL#-10.3. Значение переменной *abc* будет равно REAL#10.3.

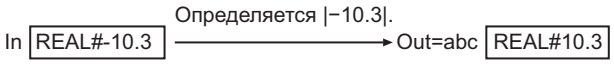
LD



ST

```
abc:=ABS(REAL#-10.3);
```

Команда ABS выводит абсолютное значение ***In***.
Определяется абсолютное значение REAL# -10.3, поэтому ***abc*** будет равно REAL#10.3.



Дополнительная информация

При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выбирайте для *Out* тип данных, поддерживающий абсолютное значение переменной *In*.
- В таблице ниже приведены значения *Out* для случаев, когда *In* является положительной бесконечностью, отрицательной бесконечностью или нечисловым значением.

Значение *In*	Значение *Out*
+ ∞	+ ∞
-∞	+ ∞
Нечисловое значение	Нечисловое значение

RadToDeg и DegToRad

RadToDeg : Преобразует вещественное число из радианов (рад) в градусы (°).
DegToRad : Преобразует вещественное число из градусов (°) в радианы (рад).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
RadToDeg	Радианы в градусы	FUN	(@)RadToDeg EN ENO In Out	Out:=RadToDeg(In);
DegToRad	Градусы в радианы	FUN	(@)DegToRad EN ENO In Out	Out:=DegToRad(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	RadToDeg: радианы DegToRad: градусы	*1
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	RadToDeg: градусы DegToRad: радианы	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														OK	OK					
Out														OK	OK					

Функция

RadToDeg

Команда RadToDeg преобразует данные для преобразования *In* из радианов (рад) в градусы (°).
Используется следующее преобразование:

$Out = In * 180/\pi$

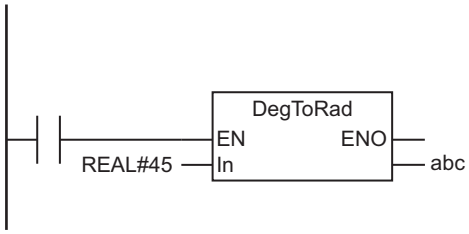
DegToRad

Команда DegToRad преобразует данные для преобразования *In* из градусов (°) в радианы (рад). Используется следующее преобразование:

$Out = In * \pi / 180$

Ниже показан пример для команды DegToRad, в котором *In* = REAL#45. Значение переменной *abc* типа REAL будет равно REAL#0.785398.

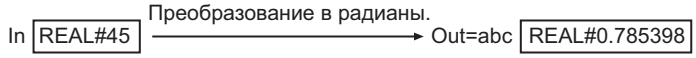
LD



ST

```
abc:=DegToRad(REAL#45);
```

Команда DegToRad преобразует значение ***In*** из градусов (°) в радианы (рад).
Угол 45° эквивалентен углу 0.785398 рад, поэтому ***abc*** будет равно REAL#0.785398.



Дополнительная информация

При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если абсолютное значение результата преобразования превышает максимальное значение типа данных *Out*, значение *Out* будет положительной или отрицательной бесконечностью.
- Если абсолютное значение результата преобразования меньше минимального значения типа данных *Out*, значение *Out* будет равно 0.
- Проследите, чтобы диапазон значений типа данных *Out* был не уже диапазона значений типа данных *In*.
- В таблице ниже приведены значения *Out* для случаев, когда *In* является положительной бесконечностью, отрицательной бесконечностью или нечисловым значением.

Значение *In*	Значение *Out*
+ ∞	+ ∞
-∞	-∞
Нечисловое значение	Нечисловое значение

- Если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

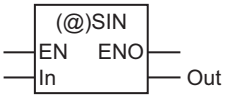
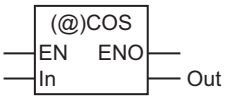
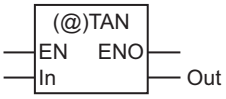
Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL

Тип данных параметра, передаваемого в *In*	Тип данных *In*
ULINT или LINT	В этом случае произойдет ошибка сборки.

SIN, COS и TAN

Эти команды выполняют тригонометрические вычисления с вещественными числами.

SIN : Вычисляет синус числа.
COS : Вычисляет косинус числа.
TAN : Вычисляет тангенс числа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SIN	Синус в радианах	FUN		Out:=SIN(In);
COS	Косинус в радианах	FUN		Out:=COS(In);
TAN	Тангенс в радианах	FUN		Out:=TAN(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Число для обработки	Вход	Число для обработки	Зависит от типа данных.	радиан	*1
Out	Результат вычисления	Выход	Результат вычисления	SIN: *2 COS: *2 TAN: Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.
*2. Диапазон допустимых значений для типа данных REAL: -1,000000e+0...1,000000e+0. Диапазон допустимых значений для типа данных LREAL: -1,00000000000000e+0...1,00000000000000e+0.

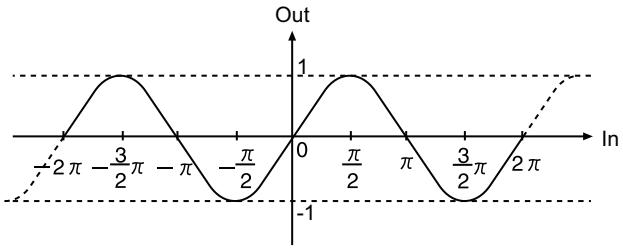
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														OK	OK					
Out														OK	OK					

Функция

Эти команды выполняют тригонометрические вычисления с вещественными числами. Число для обработки *In* является значением угла в радианах (рад).

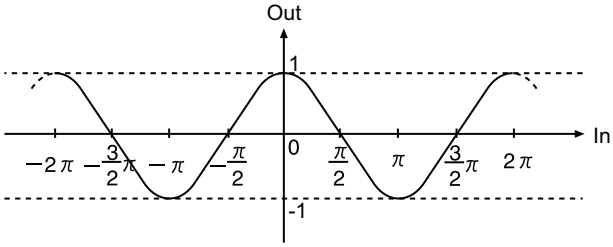
SIN

Команда SIN находит синус *In*.



COS

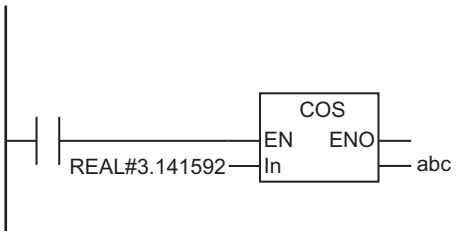
Команда COS находит косинус *In*.



● Пример для команды COS

Ниже показан пример для команды COS, в котором *In* = REAL#3.141592. Значение переменной *abc* будет равно REAL#-1.0.

LD



ST

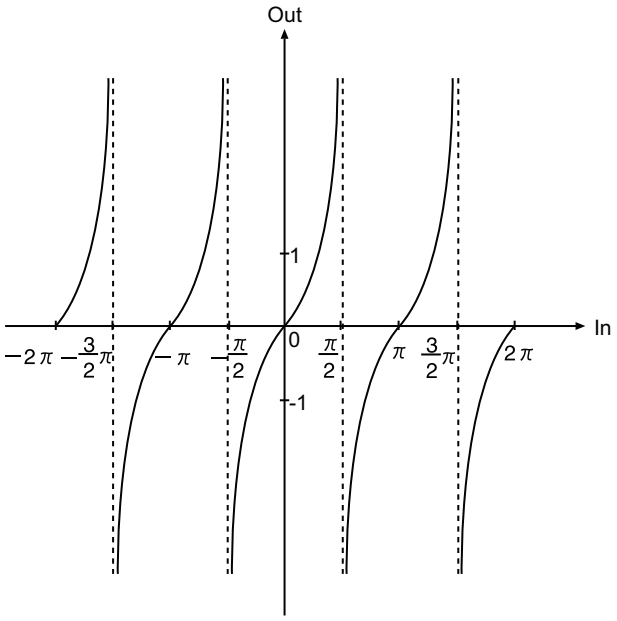
```
abc:=COS(REAL#3.141592);
```

Команда COS находит косинус ***In***.
Косинус 3.141592 равен −1.0, поэтому ***abc*** будет равно REAL# -1.0.

In REAL#3.141592 → Вычисление косинуса. → Out=abc REAL#-1.0

TAN

Команда TAN находит тангенс *In*.



Дополнительная информация

- Для преобразования радианов в градусы и наоборот используйте команды *RadToDeg u DegToRad* на стр. 2-228.
- Если на вход *In* команды TAN подано значение nπ/2 (n — целое число), значение *Out* будет положительной или отрицательной бесконечностью.
  Используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли значение *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

Меры предосторожности для обеспечения надлежащей эксплуатации

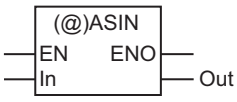
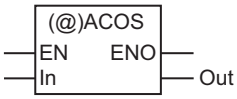
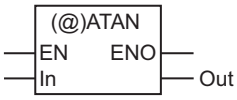
- Если значение *In* является положительной бесконечностью, отрицательной бесконечностью или нечисловым, значение *Out* является нечисловым.
- Если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

ASIN, ACOS и ATAN

Эти команды выполняют обратные тригонометрические вычисления с вещественными числами.

ASIN : Вычисляет арксинус числа (sin^{-1})
ACOS : Вычисляет арккосинус числа (cos^{-1})
ATAN : Вычисляет арктангенс числа (tan^{-1})

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ASIN	Арксинус (SIN^{-1})	FUN		Out:=ASIN(In);
ACOS	Арккосинус (COS^{-1})	FUN		Out:=ACOS(In);
ATAN	Арктангенс (TAN^{-1})	FUN		Out:=ATAN(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Число для обработки	Вход	Число для обработки	Зависит от типа данных.	---	*1
Out	Результат вычисления	Выход	Результат вычисления	ASIN: $-\pi/2..\pi/2$ ACOS: $0...\pi$ ATAN: $-\pi/2..\pi/2$	радиан	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

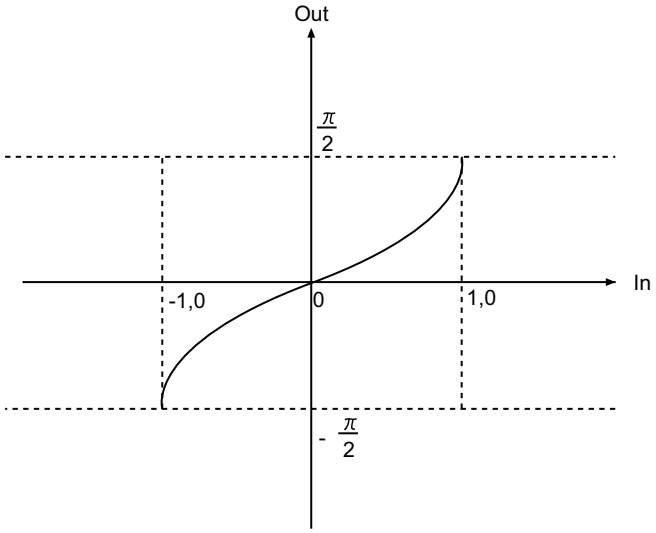
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														ОК	ОК					
Out														ОК	ОК					

Функция

Эти команды выполняют обратные тригонометрические вычисления с вещественными числами. Результат вычисления *Out* является значением угла в радианах (рад).

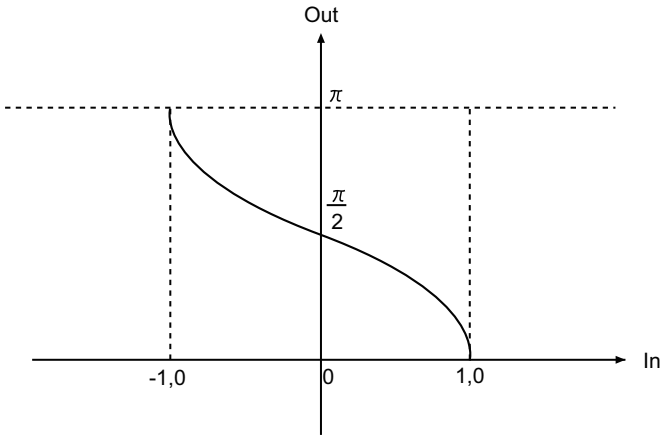
ASIN

Команда ASIN находит арксинус ($\sin^{-1}$) от *In*. *Out* принимает значение в диапазоне от $-\pi/2$ до $\pi/2$.



ACOS

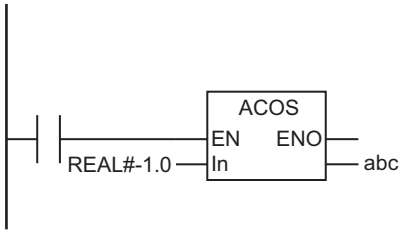
Команда ACOS находит арккосинус ($\cos^{-1}$) от *In*. *Out* принимает значение в диапазоне от 0 до π.



● Пример для команды ACOS

Ниже показан пример для команды ACOS, в котором *In* = REAL#-1.0. Значение переменной *abc* будет равно REAL#3.141592.

LD



ST

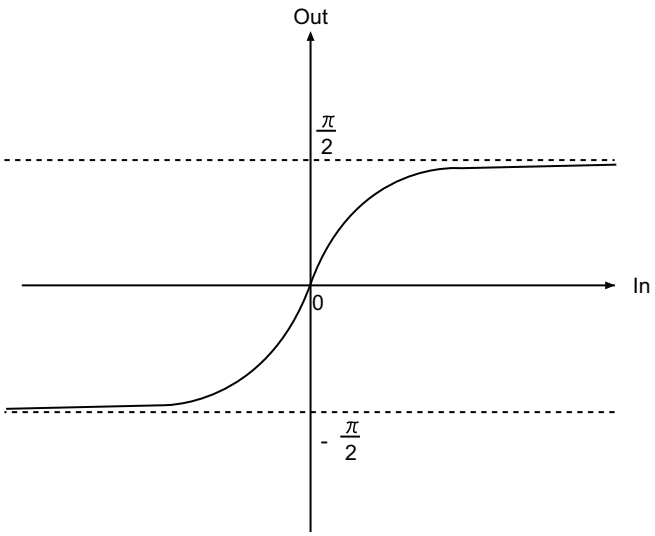
```
abc:=ACOS(REAL#-1.0);
```

Команда ACOS находит арккосинус ***In***.
Арккосинус −1.0 равен 3.141592, поэтому ***abc*** будет равно REAL#3.141592.

In REAL#-1.0 → Вычисление арккосинуса. → Out=abc REAL#3.141592

ATAN

Команда ATAN находит арктангенс ($\tan^{-1}$) от *In*. *Out* принимает значение в диапазоне от $-\pi/2$ до $\pi/2$.
Если значение *In* равно положительной бесконечности, значение *Out* равно $\pi/2$. Если значение *In* равно отрицательной бесконечности, значение *Out* равно $-\pi/2$.



Дополнительная информация

Для преобразования радианов в градусы и наоборот используйте команды *RadToDeg u DegToRad* на стр. 2-228.

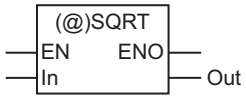
Меры предосторожности для обеспечения надлежащей эксплуатации

- Если значение *In* не находится между -1,0 и 1,0 в случае команды ASIN или ACOS, значение *Out* является нечисловым. Это также относится к случаю, когда значение *In* является положительной или отрицательной бесконечностью либо нечисловым значением.
- Если значение *In* является нечисловым в случае команды ATAN, значение *Out* является нечисловым.
- Если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

SQRT

Команда SQRT вычисляет квадратный корень из числа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SQRT	Квадратный корень	FUN		Out:=SQRT(In);

Переменные

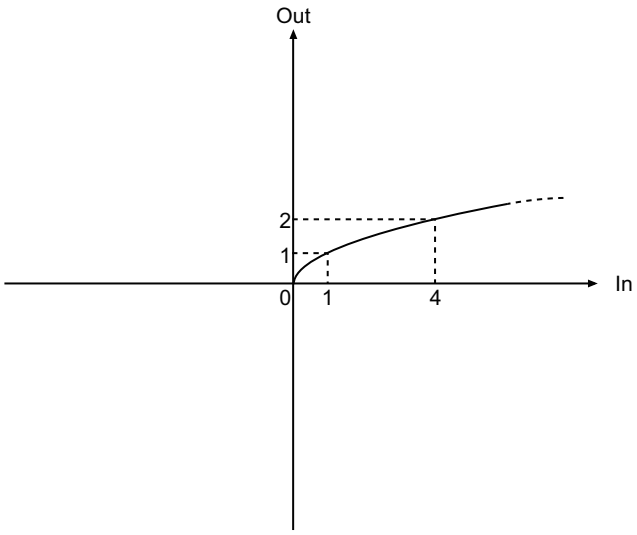
	Имя	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Число для обработки	Вход	Число для обработки	Зависит от типа данных.*1	---	*2
Out	Квадратный корень	Выход	Квадратный корень	*3	---	---

*1. Отрицательные числа исключены.
*2. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.
*3. Диапазон допустимых значений для типа данных REAL: 0,000000e+00...1,844674e+19 или положительная бесконечность. Диапазон допустимых значений для типа данных LREAL: 0,00000000000000e +000...1,34078079299425e+154 или положительная бесконечность.

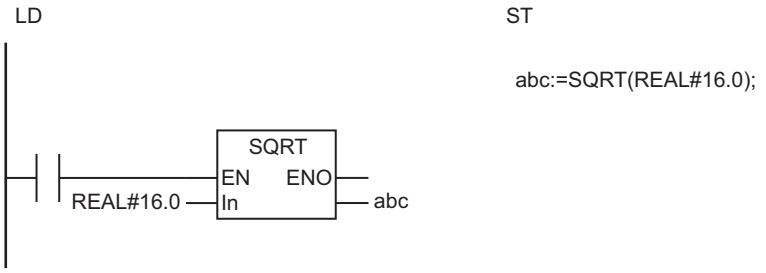
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														OK	OK					
Out														OK	OK					

Функция

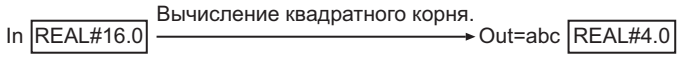
Команда SQRT находит квадратный корень из числа для обработки *In*.
Тип данных числа для обработки *In* может отличаться от типа данных квадратного корня *Out*.



В показанном ниже примере программы *In* = REAL#16.0. Значение переменной *abc* будет равно REAL#4.0.



Команда SQRT находит квадратный корень из ***In***.
Квадратный корень из 16.0 равен 4.0, поэтому ***abc*** будет равно REAL#4.0.



Дополнительная информация

Используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли значение *Out* положительной бесконечностью.

Меры предосторожности для обеспечения надлежащей эксплуатации

- В таблице ниже приведены значения *Out* для случаев, когда значение *In* не является положительным числом.

Значение *In*	Значение *Out*
Отрицательное число	Нечисловое значение
0	0
+ ∞	+ ∞
-∞	Нечисловое значение
Нечисловое значение	Нечисловое значение

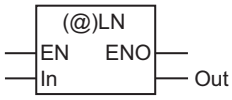
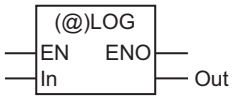
- Если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

LN и LOG

Эти команды вычисляют логарифм вещественного числа.

LN : Вычисляет натуральный логарифм числа.
LOG : Вычисляет десятичный логарифм числа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LN	Натуральный логарифм	FUN		Out:=LN(In);
LOG	Десятичный логарифм	FUN		Out:=LOG(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Число для обработки	Вход	Число для обработки	Зависит от типа данных.*1	---	*2
Out	Логарифм	Выход	Логарифм	*3	---	---

*1. Отрицательные числа исключены.
*2. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.
*3. Диапазон допустимых значений зависит от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Диапазон допустимых значений* на стр. 2-242.

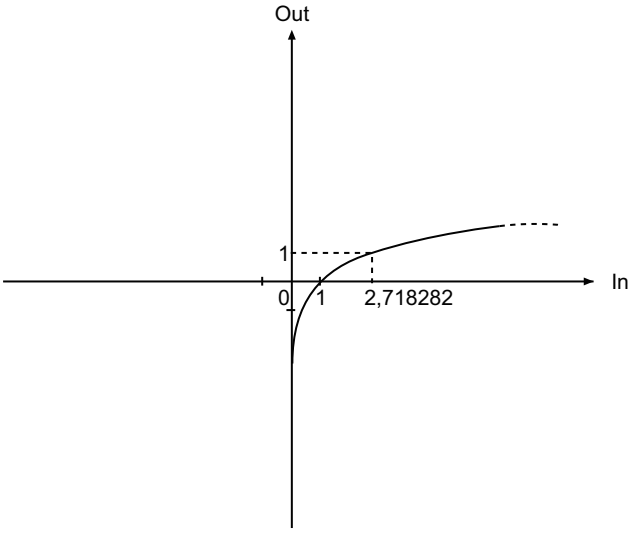
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														ОК	ОК					
Out														ОК	ОК					

Функция

Эти команды находят логарифм вещественного числа.

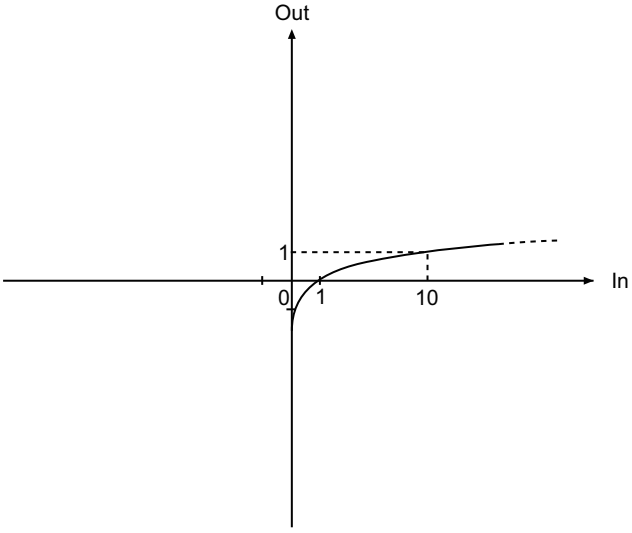
LN

Команда LN находит натуральный логарифм (логарифм по основанию «e», где e = 2,718282).



LOG

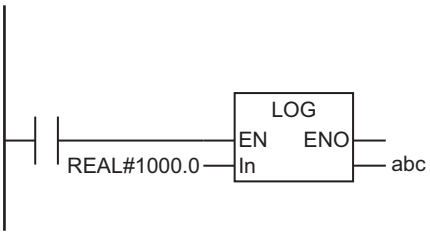
Команда LOG находит десятичный логарифм (логарифм по основанию 10).



● Пример для команды LOG

Ниже показан пример для команды LOG, в котором *In* = REAL#1000.0. Значение переменной *abc* будет равно REAL#3.0.

LD



ST

```
abc:=LOG(REAL#1000.0);
```

Команда LOG вычисляет логарифм по основанию 10 от вещественного числа.
Логарифм по основанию 10 от 1000.0 равен 3.0, поэтому ***abc*** будет равно REAL#3.0.

Диапазон допустимых значений

В следующих таблицах приведены диапазоны допустимых значений для команд LN и LOG.

● Диапазоны допустимых значений для команды LN

Тип данных *In*	Тип данных *Out*	Диапазон допустимых значений
REAL	REAL	-8,73365448e+1...8,87228390e+1 или -∞/+∞
REAL	LREAL	-8,7336544750000000e+1...8,8722839050000000e+1 или -∞/+∞
LREAL	REAL	-7,08384950e+2...7,09782712e+2 или -∞/+∞
LREAL	LREAL	-7,0838495021978327e+1...7,0978271289338399e+2 или -∞/+∞

● Диапазоны допустимых значений для команды LOG

Тип данных *In*	Тип данных *Out*	Диапазон допустимых значений
REAL	REAL	-3,79297795e+1...3,85318394e+1 или -∞/+∞
REAL	LREAL	-3,7929779453965430e+1...3,8531839419564961e+1 или -∞/+∞
LREAL	REAL	-3,07652656e+2...3,08254716e+2 или -∞/+∞
LREAL	LREAL	-3,0765265556858878e+2...3,0825471555991674e+2 или -∞/+∞

Дополнительная информация

При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

Меры предосторожности для обеспечения надлежащей эксплуатации

- В таблице ниже приведены значения *Out* для случаев, когда значение *In* не является положительным числом.

Значение *In*	Значение *Out*
Отрицательное число	Нечисловое значение
0	-∞
+ ∞	+ ∞
-∞	Нечисловое значение
Нечисловое значение	Нечисловое значение

- Если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

EXP

Команда EXP вычисляет значение экспоненциальной функции.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EXP	Экспонента	FUN	(@)EXP EN ENO In Out	Out:=EXP(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Показатель степени	Вход	Показатель степени	Зависит от типа данных.	---	*1
Out	Результат вычисления	Выход	Результат вычисления	Зависит от типа данных.*2	---	---

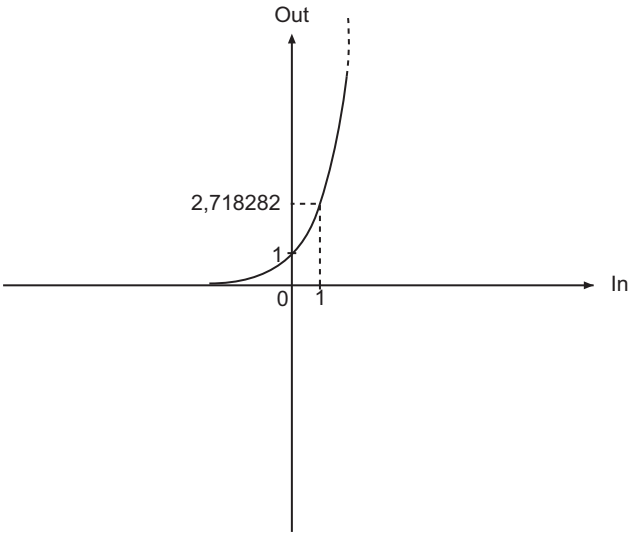
*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

*2. Отрицательные числа исключены.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														OK	OK					
Out														OK	OK					

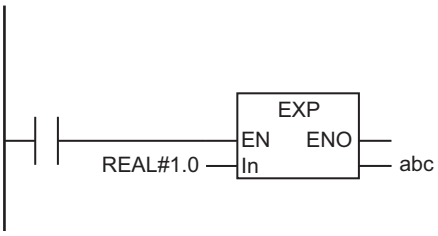
Функция

Команда EXP возводит основание натурального логарифма («e») в степень *In*.



В показанном ниже примере программы *In* = REAL#1.0. Значение переменной *abc* будет равно REAL#2.718282.

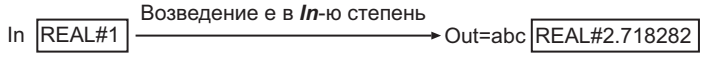
LD



ST

```
abc:=EXP(REAL#1.0);
```

Команда EXP возвращает натуральный логарифм от ***In*** по основанию e.
Значение e^1 = 2.718282, поэтому ***abc*** будет равно REAL#2.718282.



Дополнительная информация

- Для возведения в степень других чисел (не «e») используйте команду *EXPT (**)* на стр. 2-247.
- При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

Меры предосторожности для обеспечения надлежащей эксплуатации

- В таблице ниже приведены значения *Out* для случаев, когда *In* равно 0,0 либо является положительной бесконечностью, отрицательной бесконечностью или нечисловым значением.

Значение *In*	Значение *Out*	
	NX1P2	Другие модели
0,0	1,0	1,0
+ ∞	+ ∞	Нечисловое значение
-∞	0,0	Нечисловое значение

Значение *In*	Значение *Out*	
	NX1P2	Другие модели
Нечисловое значение	Нечисловое значение	Нечисловое значение

- Если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

EXPT (**)

Команда EXPT (**) возводит одно вещественное число в степень, равную другому вещественному числу.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EXPT (**)	Возведение в степень	FUN	(@)EXPT EN ENO In Out Pwr (@)** EN ENO In Out Pwr	Out:=EXPT(In, Pwr); Out:=In ** Pwr;

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Основание степени	Вход	Основание степени (например, 5 для 5^2)	Зависит от типа данных.	---	*1
Pwr	Показатель степени		Показатель степени (например, 2 для 5^2)			
Out	Результат вычисления	Выход	Результат вычисления	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

Команда языка LD и команда EXPT языка ST

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														OK	OK					
Pwr														OK	OK					
Out														OK	OK					

Оператор ** языка ST

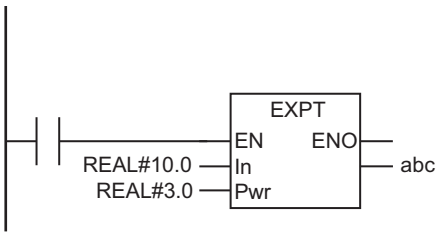
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
Pwr						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
Out						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					

Функция

Команда EXPT (**) возводит значение основания *In* в степень *Pwr* (т. е. вычисляет In^{Pwr}).

Ниже показан пример, в котором переменные *In* и *Pwr* имеют значения REAL#10.0 и REAL#3.0 соответственно. Значение переменной *abc* будет равно REAL#1000.0.

LD



ST

```
abc:=EXPT(REAL#10.0, REAL#3.0);
```

Команда EXPT возводит ***In*** в степень ***Pwr***.
$10.0^{3.0}$ = 1000.0, поэтому ***abc*** будет равно REAL#1000.0.

In REAL#10.0
Pwr REAL#3.0
→ Вычисление ***In*** в степени ***Pwr*** ($10.0^{3.0}$). → Out=abc REAL#1000.0

Различия между языками LD и ST

Характеристики данной команды зависят от используемого языка программирования (LD или ST). Характеристики при использовании оператора ** в программе на языке ST отличаются от характеристик при использовании команд EXPT и ** в программе на языке LD и при использовании функции EXPT в программе на языке ST. В следующей таблице приводятся различающиеся характеристики.

Параметр	Функции EXPT в программах на языке LD и ST	Оператор ** в программе на языке ST
Наличие *EN* и *ENO*	Есть	Нет

Параметр	Функции EXPT в программах на языке LD и ST	Оператор ** в программе на языке ST
Разрядность вычислений, когда *In* и *Pwr* являются целыми числами	32 или 64*1	64*2

*1. Операции выполняются с типом данных REAL или LREAL, в зависимости от того, что из них меньше. Например, если операндами являются значения типа SINT и DINT, длина обрабатываемых значений приводится к типу данных LREAL, т. е. вычисления выполняются для 64-битных значений.

*2. Выполняются 64-битные вычисления. Например, если одно значение типа SINT возводится в степень, равную другому значению типа SINT, операции выполняются над 64-битными числами.

Дополнительная информация

- Для возведения основания натурального логарифма «e» в степень используйте команду *EXP* на стр. 2-244.
- При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если абсолютное значение результата вычислений меньше минимально возможного значения вещественного числа, значение *Out* будет равно 0.

  Пример: $(1{,}175494e\text{-}38)^2 \rightarrow 0$
- Потеря значимости или переполнение при вычислении выражения с оператором ** не приводят к возникновению ошибки.
- Если при вычислении выражения с оператором ** происходит потеря значимости или переполнение, результат вычислений может отличаться от ожидаемого. Чтобы потери значимости или переполнения не возникало, выбирайте для входных и выходных параметров типы данных с достаточным запасом разрядности.
- Для команд EXPT и ** в программе на языке LD и для функции EXPT в программе на языке ST: если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

- Если для модуля ЦПУ NX701 или модуля ЦПУ серии NJ в области Select Device (Выбор устройства) диалогового окна Project Properties (Свойства проекта) в Sysmac Studio для параметра **Version (Версия)** выбрано значение **1.15** или меньше, то при использовании оператора ** целочисленные переменные обрабатываются в вычислениях как переменные вещественного типа, даже если они указаны в качестве операндов.

  Так как результат вычислений округляется до целого значения, он может быть неверным из-за ошибки округления.

  Используйте команды EXPT и TO_** (Группа преобразования целых чисел) вместе для округления значения до целого числа.

  Пример: TO_INT (EXPT(X,Y))

Комбинации значений *In* и *Pwr*

В следующей таблице приведены значения *Out* для различных комбинаций значений *In* и *Pwr*.

● Функция *EXPT* для устройства, не являющегося модулем ЦПУ NX1P2

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="10">In</th></tr>
<tr><th>+ ∞</th><th>1...+∞</th><th>1</th><th>0...1</th><th>0</th><th>-1...0</th><th>-1</th><th>-1...-∞</th><th>-∞</th><th>Нечисловое значение</th></tr>
<tr><td rowspan="11">Pwr</td><td>+ ∞</td><td>+ ∞</td><td>+ ∞</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>+ ∞</td><td>1</td><td>Нечисловое значение</td></tr>
<tr><td>Положительное четное число</td><td rowspan="3">+ ∞</td><td rowspan="3">Число *1*2</td><td rowspan="3">1</td><td rowspan="3">Число *1*2</td><td rowspan="3">0</td><td>Число *1*2</td><td>1</td><td>Число *1*2</td><td rowspan="3">+ ∞</td><td rowspan="3">Нечисловое значение</td></tr>
<tr><td>Положительное нечетное число</td><td>Число *2*3</td><td>-1</td><td>Число *2*3</td></tr>
<tr><td>Положительное десятичное число</td><td colspan="3">Нечисловое значение</td></tr>
<tr><td>0</td><td>1</td><td colspan="3">1</td><td>1</td><td colspan="3">1</td><td>1</td><td>1</td></tr>
<tr><td>Отрицательное четное число</td><td rowspan="3">0</td><td rowspan="3">Число *1*2</td><td rowspan="3">1</td><td rowspan="3">Число *1*2</td><td rowspan="3">+ ∞</td><td>Число *1*2</td><td>1</td><td>Число *1*2</td><td rowspan="3">0</td><td rowspan="3">Нечисловое значение</td></tr>
<tr><td>Отрицательное нечетное число</td><td>Число *2*3</td><td>-1</td><td>Число *2*3</td></tr>
<tr><td>Отрицательное десятичное число</td><td colspan="3">Нечисловое значение</td></tr>
<tr><td>-∞</td><td>0</td><td>0</td><td>1</td><td>+ ∞</td><td>+ ∞</td><td>+ ∞</td><td>1</td><td>0</td><td>0</td><td>Нечисловое значение</td></tr>
<tr><td>Нечисловое значение</td><td>1</td><td>Нечисловое значение</td><td>1</td><td>Нечисловое значение</td><td>1</td><td colspan="3">Нечисловое значение</td><td>1</td><td>Нечисловое значение
1</td></tr>
</table>

*1. Если результат вычисления выходит за диапазон допустимых значений типа данных *Out*, значение *Out* будет равно +∞.

*2. Если результат вычисления слишком близок к *0* и поэтому не может быть выражен типом данных *Out* или если результат вычисления является денормализованным числом, значение *Out* будет равно *0*.

*3. Если результат вычисления выходит за диапазон допустимых значений типа данных *Out*, значение *Out* будет равно -∞.

● Функция *EXPT* для модуля ЦПУ NX1P2

		In									
		+ ∞	**1...+∞**	**1**	**0...1**	**0**	**-1...0**	**-1**	**-1...-∞**	**-∞**	**Нечисловое значение**
Pwr	**+ ∞**	+ ∞	+ ∞	1	0	1	0	1	+ ∞	1	Нечисловое значение
	Положительное четное число	+ ∞	Число *1*2	1	Число *1*2	0	Число *1*2	1	Число *1*2	+ ∞	Нечисловое значение
	Положительное нечетное число						Число *2*3	-1	Число *2*3	-∞	
	Положительное десятичное число						Нечисловое значение			+ ∞	
	0	1	1			1	1			1	1
	Отрицательное четное число	0	Число *1*2	1	Число *1*2	+ ∞	Число *1*2	1	Число *1*2	0	Нечисловое значение
	Отрицательное нечетное число						Число *2*3	-1	Число *2*3	-0	
	Отрицательное десятичное число						Нечисловое значение			0	
	-∞	0	0	1	+ ∞	+ ∞	+ ∞	1	0	0	Нечисловое значение
	Нечисловое значение	1	Нечисловое значение	1	Нечисловое значение	1	Нечисловое значение			1	Нечисловое значение 1

*1. Если результат вычисления выходит за диапазон допустимых значений типа данных *Out*, значение *Out* будет равно +∞.

*2. Если результат вычисления слишком близок к *0* и поэтому не может быть выражен типом данных *Out* или если результат вычисления является денормализованным числом, значение *Out* будет равно *0*.

*3. Если результат вычисления выходит за диапазон допустимых значений типа данных *Out*, значение *Out* будет равно -∞.

● Оператор **

		In									
		+ ∞	**1...+∞**	**1**	**0...1**	**0**	**-1...0**	**-1**	**-1...-∞**	**-∞**	**Нечисловое значение**
Pwr	**+ ∞**	+ ∞	+ ∞	1	0	1	0	1	+ ∞	1	Нечисловое значение
	Положительное четное число	+ ∞	Число *1*2*3	1	Число *1*2	0	Число *1*2	1	Число *1*2*3	+ ∞	Нечисловое значение
	Положительное нечетное число						Число *2*4	-1	Число *2*3*4	-∞	
	Положительное десятичное число						Нечисловое значение			+ ∞	
	0	1	1			1	1			1	1
	Отрицательное четное число	0	Число *1*2	1	Число *1*2	+∞ *5	Число *1*2	1	Число *1*2	0	Нечисловое значение
	Отрицательное нечетное число						Число *2*4	-1	Число *2*4	-0	
	Отрицательное десятичное число						Нечисловое значение			0	
	-∞	0	0	1	+ ∞	+ ∞	+ ∞	1	0	0	Нечисловое значение
	Нечисловое значение	1	Нечисловое значение			1	Нечисловое значение			1	Нечисловое значение 1

*1. Если результат вычисления выходит за диапазон допустимых значений типа данных *Out*, значение *Out* будет равно +∞.

*2. Если результат вычисления слишком близок к *0* и поэтому не может быть выражен типом данных *Out* или если результат вычисления является денормализованным числом, значение *Out* будет равно *0*.

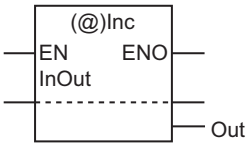
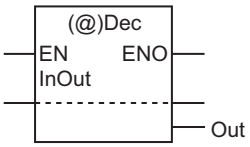
*3. Если *In* и *Pwr* являются целыми значениями, то *Out* содержит неопределенное значение, когда результат вычисления превышает диапазон допустимых значений типа данных *Out*.

*4. Если результат вычисления выходит за диапазон допустимых значений типа данных *Out*, значение *Out* будет равно -∞.

*5. Если *In* и *Pwr* являются целыми значениями, то *Out* содержит неопределенное значение.

Inc и Dec

Inc : Увеличивает целочисленное значение на единицу.
Dec : Уменьшает целочисленное значение на единицу.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Inc	Увеличение на единицу	FUN		Inc(InOut);
Dec	Уменьшение на единицу	FUN		Dec(InOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
InOut	Целевые данные	Вход-выход	Целевые данные	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК							
Out	ОК																			

Функция

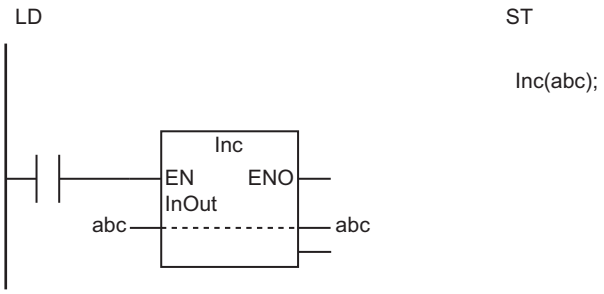
Inc

Команда Inc увеличивает на единицу целевые данные *InOut*. Если значение результата превышает максимальное значение *InOut*, оно возвращается к минимальному значению.

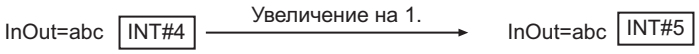
● Пример для команды Inc

Ниже показан пример для команды Inc, в котором в параметр *InOut* передается переменная *abc*.

Если переменная *abc* содержит значение INT#4, то после выполнения команды ее значение станет равно INT#5.



Команда Inc увеличивает ***InOut*** на 1.
Если ***abc*** равно INT#4, то после выполнения команды ***abc*** станет равно INT#5.



Dec

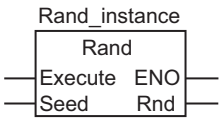
Команда Dec уменьшает на единицу целевые данные *InOut*. Если значение результата становится меньше минимального значения *InOut*, оно возвращается к максимальному значению.

Меры предосторожности для обеспечения надлежащей эксплуатации

При использовании этих команд в программе на языке ST возвращаемое значение *Out* не используется.

Rand

Команда Rand генерирует псевдослучайные числа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Rand	Случайное число	FB		Rand_instance(Execute, Seed, Rnd);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Seed	Начальное число для генерации случайных чисел	Вход	Начальное число для генерации случайных чисел 0: не указано.	Зависит от типа данных.	---	*1
Rnd	Случайное число	Выход	Случайное число	*2	---	---

*1. Если входной параметр будет опущен, значение будет равно 0. Оно не будет равно значению, которое указано для атрибута начального значения (Initial Value).

*2. 0,00000000000000e+0...1,00000000000000e+0

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Seed							OK													
Rnd															OK					

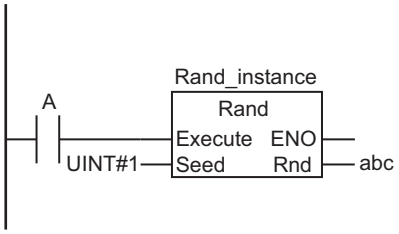
Функция

Команда Rand указывает случайное число *Rnd*. При каждом выполнении команды переменная *Rnd* принимает новое значение, отличающееся от предыдущих.
Последовательность случайных чисел зависит от заданного начального значения *Seed*. Если значение *Seed* не меняется, то после включения питания каждый раз генерируется один и тот же ряд случайных чисел. Это позволяет генерировать воспроизводимый ряд случайных чисел.

Если же значение *Seed* равно 0, то генерируются невоспроизводимые случайные числа. Если не нужно, чтобы при включении питания каждый раз генерировался один и тот же ряд случайных чисел, передайте в *Seed* значение 0.

В показанном ниже примере программы *Seed* = UINT#1. Так как *Seed* не равно 0, то генерируются воспроизводимые случайные числа.

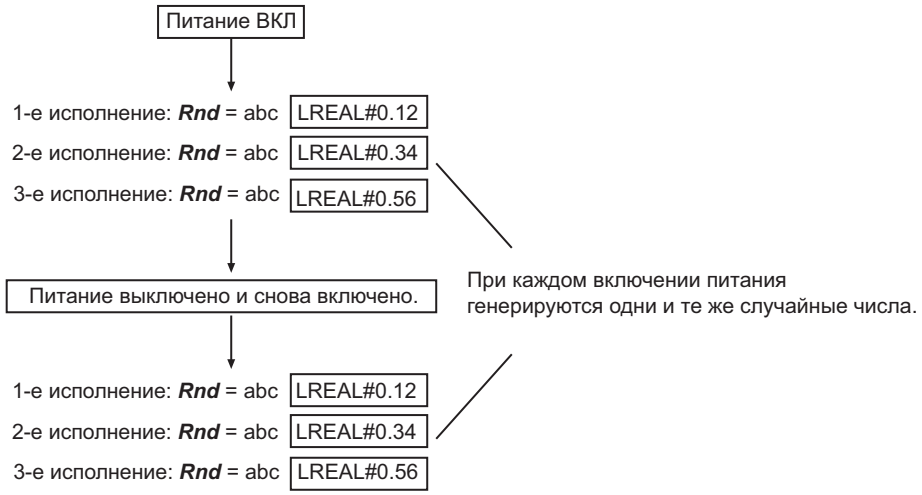
LD



ST

```
Rand_instance(A, UINT#1, abc);
```

Команда Rand генерирует повторяющийся ряд случайных чисел.



* Значения случайных чисел выше приведены в качестве примера. Фактические значения будут другими.

Дополнительная информация

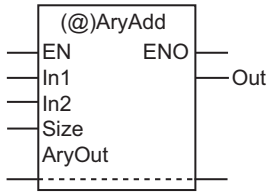
Значение *Rnd* представляет собой вещественное число в диапазоне от 0 до 1. Чтобы генерировать случайные числа в определенном диапазоне, используйте показанный ниже прием.

(Пример) Следующая формула генерирует случайные числа в диапазоне от 100 до 200.

```
Rand_instance(A, UINT#1, abc);
Random number:=LREAL_TO_INT((200.0-100.0)*abc)+100;
```

AryAdd

Команда AryAdd складывает соответствующие элементы двух массивов.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryAdd	Сложение массивов	FUN		AryAdd(In1, In2, Size, AryOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1[] (массив), In2[] (массив)	Массив для обработки	Вход	Массив для обработки	Зависит от типа данных.	---	*1
Size	Количество элементов для обработки		Количество элементов для обработки			1
AryOut[] (массив)	Массив с результатами вычислений	Вход-выход	Массив с результатами вычислений	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

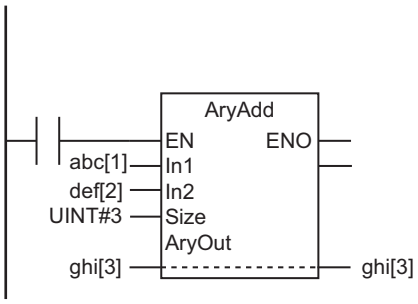
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
In2[] (массив)	Должен быть массивом с тем же типом данных, что и In1[].																			
Size							ОК													
AryOut[] (массив)	Должен быть массивом с тем же типом данных, что и In1[].																			
Out	ОК																			

Функция

Команда AryAdd попарно складывает элементы двух массивов In1[] и In2[], начиная с элементов In1[0] и In2[0] соответственно. Количество элементов задается параметром *Size*. Результат сложения каждой пары выдается в соответствующий элемент массива AryOut[] (массив результатов вычислений).

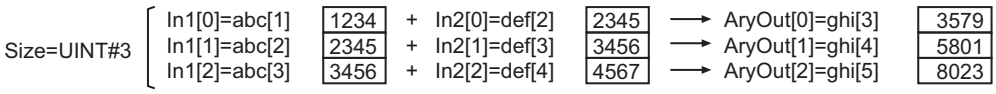
В показанном ниже примере программы *Size* = UINT#3.

LD



ST

```
AryAdd(abc[1], def[2], UINT#3, ghi[3]);
```

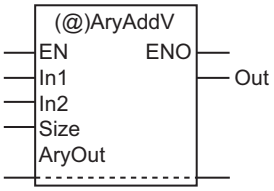


Меры предосторожности для обеспечения надлежащей эксплуатации

- Для In1[], In2[] и AryOut[] следует использовать один и тот же тип данных. При использовании разных типов данных произойдет ошибка сборки.
- Если результаты вычислений превысят диапазон допустимых значений AryOut[], результаты будут содержать недопустимые значения. Это не приведет к ошибке. Данные в области памяти, смежной с такими элементами, повреждены не будут.
- Если *Size* = 0, значения в массиве AryOut[] не изменяются.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение *Size* превышает количество элементов массива In1[], In2[] или AryOut[].

AryAddV

Команда AryAddV прибавляет одно и то же значение к указанным элементам массива.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryAddV	Прибавление значения к массиву	FUN		AryAddV(In1, In2, Size, AryOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1[] (массив)	Массив для сложения	Вход	Массив для сложения	Зависит от типа данных.	---	*1
In2	Добавляемое значение		Добавляемое значение			
Size	Количество элементов		Количество элементов In1[], к которым должно быть прибавлено значение			1
AryOut[] (массив)	Массив с результатами сложения	Вход-выход	Массив с результатами сложения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

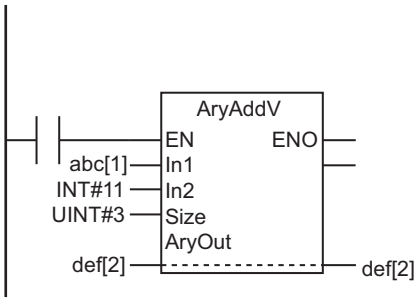
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
In2	Тип данных должен быть таким же, как у In1[].																			
Size							ОК													
AryOut[] (массив)	Тип данных должен быть таким же, как у In1[].																			
Out	ОК																			

Функция

Команда AryAddV прибавляет значение *In2* (добавляемое значение) к каждому из *Size* элементов массива In1[] (массив для сложения), начиная с элемента In1[0]. Результат сложения для каждого элемента выдается в соответствующий элемент массива AryOut[] (массив результатов сложения).

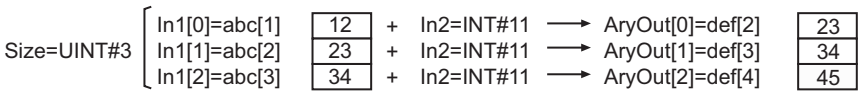
Ниже показан пример, в котором переменные *In2* и *Size* имеют значения INT#11 и UINT#3 соответственно.

LD



ST

```
AryAddV(abc[1], INT#11, UINT#3, def[2]);
```

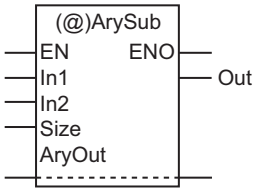


Меры предосторожности для обеспечения надлежащей эксплуатации

- Для In1[], *In2* и AryOut[] следует использовать один и тот же тип данных. В противном случае произойдет ошибка сборки.
- Если результаты сложения выйдут за диапазон допустимых значений AryOut[], элементы AryOut[] будут содержать недопустимые значения. Это не приведет к ошибке. Данные в области памяти, смежной с такими элементами, повреждены не будут.
- Если *Size* = 0, значения в массиве AryOut[] не изменяются.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение *Size* превышает количество элементов массива In1[] или AryOut[].

ArySub

Команда ArySub вычитает соответствующие элементы двух массивов.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ArySub	Вычитание массивов	FUN		ArySub(In1, In2, Size, AryOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1[] (массив)	Массив уменьшаемых	Вход	Массив уменьшаемых	Зависит от типа данных.	---	*1
In2[] (массив)	Массив вычитаемых		Массив вычитаемых			
Size	Количество элементов		Количество элементов для вычитания			1
AryOut[] (массив)	Массив с результатами вычитания	Вход-выход	Массив с результатами вычитания	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

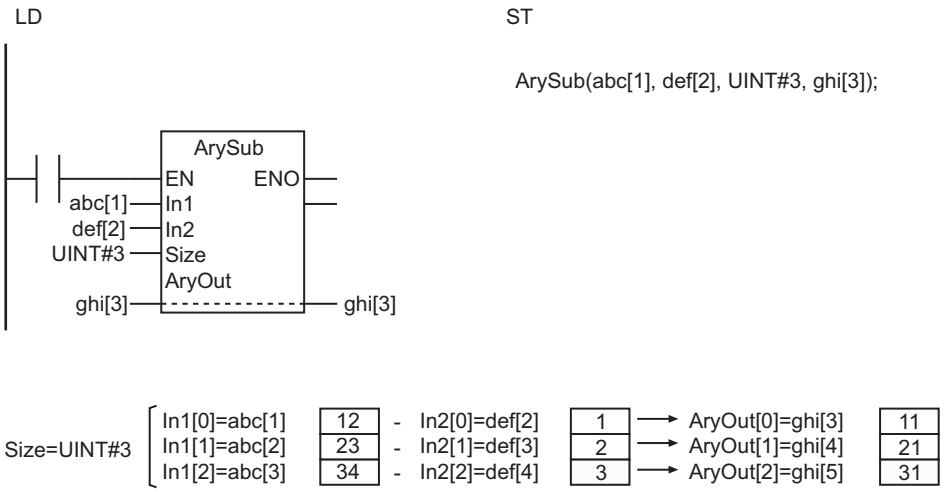
*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
In2[] (массив)	Тип данных должен быть таким же, как у In1[].																			
Size							ОК													
AryOut[] (массив)	Тип данных должен быть таким же, как у In1[].																			
Out	ОК																			

Функция

Команда ArySub вычитает каждый из *Size* элементов массива In2[] (массив вычитаемых) из соответствующего элемента массива In1[] (массив уменьшаемых). Результат вычитания для каждой пары выдается в соответствующий элемент массива AryOut[] (массив результатов вычитания).

В показанном ниже примере программы *Size* = UINT#3.

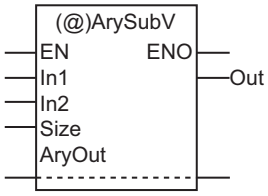


Меры предосторожности для обеспечения надлежащей эксплуатации

- Для In1[], In2[] и AryOut[] следует использовать один и тот же тип данных. При использовании разных типов данных произойдет ошибка сборки.
- Если результаты вычитания выйдут за диапазон допустимых значений AryOut[], элементы AryOut[] будут содержать недопустимые значения. Это не приведет к ошибке. Данные в области памяти, смежной с такими элементами, повреждены не будут.
- Если *Size* = 0, значения в массиве AryOut[] не изменяются.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение *Size* превышает количество элементов массива In1[], In2[] или AryOut[].

ArySubV

Команда ArySubV вычитает одно и то же значение из указанных элементов массива.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ArySubV	Вычитание значения из массива	FUN		ArySubV(In1, In2, Size, AryOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1[] (массив)	Массив уменьшаемых	Вход	Массив уменьшаемых	Зависит от типа данных.	---	*1
In2	Вычитаемое		Вычитаемое			
Size	Количество элементов		Количество элементов In1[], от которых должно быть отнято значение			1
AryOut[] (массив)	Массив с результатами вычитания	Вход-выход	Массив с результатами вычитания	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

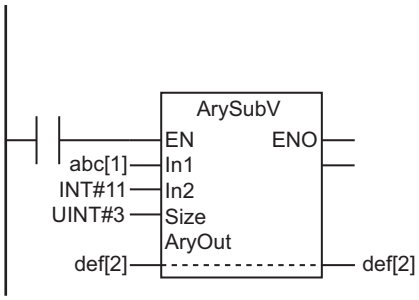
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
In2	Тип данных должен быть таким же, как у In1[].																			
Size							ОК													
AryOut[] (массив)	Тип данных должен быть таким же, как у In1[].																			
Out	ОК																			

Функция

Команда ArySubV отнимает значение *In2* (вычитаемое) от значения каждого из *Size* элементов массива In1[] (массив уменьшаемых), начиная с элемента In1[0]. Результат вычитания для каждой пары выдается в соответствующий элемент массива AryOut[] (массив результатов вычитания).

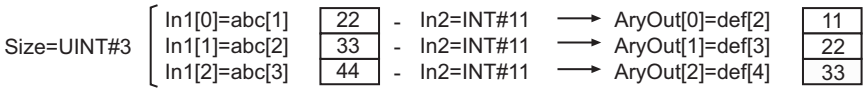
Ниже показан пример, в котором переменные *In2* и *Size* имеют значения INT#11 и UINT#3 соответственно.

LD



ST

```
ArySubV(abc[1], INT#11, UINT#3, def[2]);
```

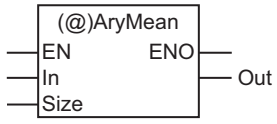


Меры предосторожности для обеспечения надлежащей эксплуатации

- Для In1[], *In2* и AryOut[] следует использовать один и тот же тип данных. В противном случае произойдет ошибка сборки.
- Если результаты вычитания выйдут за диапазон допустимых значений AryOut[], элементы AryOut[] будут содержать недопустимые значения. Это не приведет к ошибке. Данные в области памяти, смежной с такими элементами, повреждены не будут.
- Если *Size* = 0, значения в массиве AryOut[] не изменяются.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение *Size* превышает количество элементов массива In1[] или AryOut[].

AryMean

Команда AryMean вычисляет среднее значение элементов массива.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryMean	Среднее значение массива	FUN		Out := AryMean(In, Size);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для обработки	Вход	Массив для обработки	Зависит от типа данных.	---	*1
Size	Количество элементов для обработки		Количество элементов массива In[]			1
Out	Результат вычисления	Выход	Результат вычисления	Зависит от типа данных.	---	---

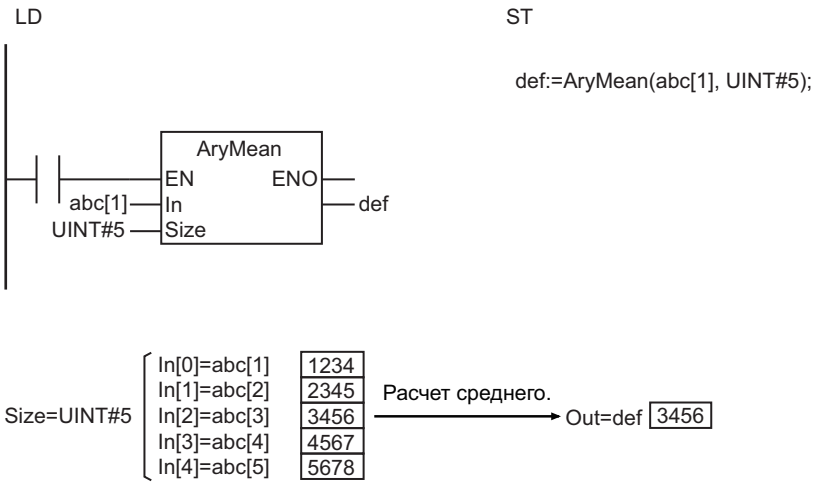
*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК				
Size							ОК													
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК				

Функция

Команда AryMean вычисляет среднее значение элементов массива In[] (массив для обработки), начиная с элемента In[0]. Число элементов для усреднения задается параметром *Size*.

В показанном ниже примере программы *Size* = UINT#5.



Меры предосторожности для обеспечения надлежащей эксплуатации

- Сведения о результатах вычислений в случаях, когда значение In[] представляет собой положительную бесконечность, отрицательную бесконечность или нечисловые данные, см. в описаниях команд *ADD (+)* на стр. 2-195, *SUB (-)* на стр. 2-204, *MUL (*)* на стр. 2-212 и *DIV (/)* на стр. 2-220.
- Если In[] и *Out* являются целыми числами, среднее значение усекается до целого числа.
- Если типы данных In[] и *Out* отличаются, проследите, чтобы диапазон допустимых значений *Out* вмещал в себя диапазон допустимых значений In[].
- Если результат вычислений превысит диапазон допустимых значений *Out*, *Out* будет содержать недопустимое значение. Это не приведет к ошибке.
- Если некоторое промежуточное значение в процессе вычислений превысит диапазон допустимых значений In[], *Out* будет содержать недопустимое значение. Это не приведет к ошибке.
- Если значение *Size* равно 0, то значение *Out* также равно 0.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

  a) Значение *Size* превышает количество элементов массива In[].

ArySD

Команда ArySD вычисляет среднеквадратическое отклонение элементов массива.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ArySD	Среднеквадратическое отклонение элементов массива	FUN	(@)ArySD EN ENO In Out Size	Out:=ArySD(In, Size);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для обработки	Вход	Массив для обработки	Зависит от типа данных.	---	*1
Size	Количество элементов		Количество элементов In[] для преобразования			2
Out	Среднеквадратическое отклонение	Выход	Среднеквадратическое отклонение	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)														OK	OK					
Size							OK													
Out														OK	OK					

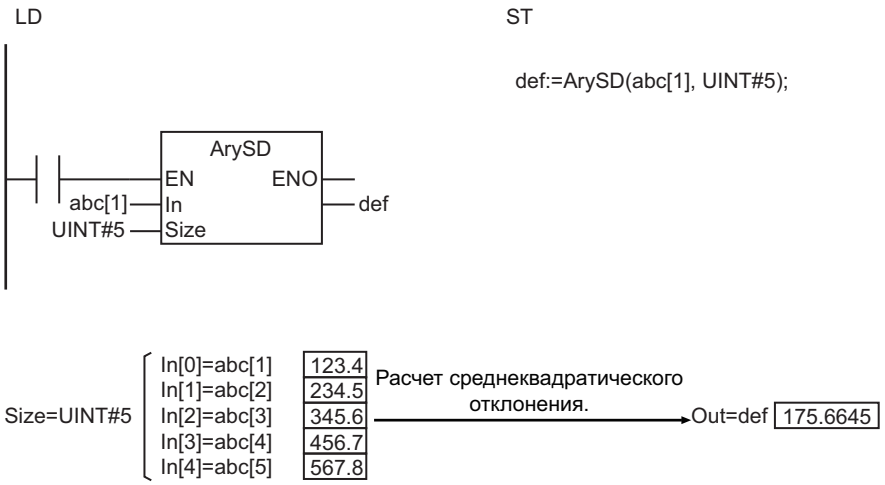
Функция

Команда ArySD вычисляет среднеквадратическое отклонение элементов массива In[] (массив для обработки), начиная с элемента In[0]. Число элементов для обработки задается параметром *Size*.

$$\text{Среднеквадратическое отклонение} = \sqrt{\frac{\sum_{i}(\text{In}[i]-\text{InM})^2}{\text{Size}-1}}$$

i: индекс ***In[]***, от 0 до ***Size*** - 1
InM: среднее значение от ***In[0]...In[Size - 1]***

В показанном ниже примере программы *Size* = UINT#5.

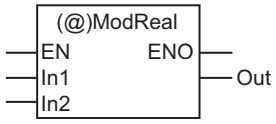


Меры предосторожности для обеспечения надлежащей эксплуатации

- Если значение *Size* равно 0 или 1, то значение *Out* равно 0.
- Если некоторое промежуточное значение в процессе вычислений превысит диапазон допустимых значений In[], *Out* будет содержать недопустимое значение. Это не приведет к ошибке.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Size* превышает количество элементов массива In[].

ModReal

Команда ModReal вычисляет остаток от деления вещественных чисел.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ModReal	Вещественное деление по модулю	FUN		Out:=ModReal(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Делимое	Вход	Делимое	Зависит от типа данных.	---	*1
In2	Делитель		Делитель			
Out	Остаток	Выход	Остаток	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1														OK	OK					
In2														OK	OK					
Out														OK	OK					

Функция

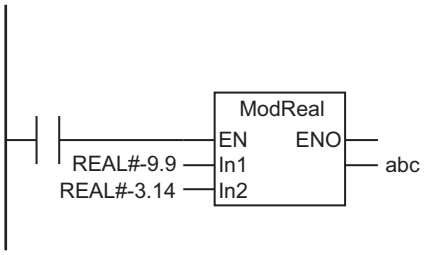
Команда ModReal делит делимое *In1* на делитель *In2* и определяет остаток от деления.

Эта команда выполняет вычисление по следующей формуле:
Out = *In1* - (*In1*/*In2*)**In2*
Дробные части при выполнении деления отбрасываются.
В таблице ниже приведены значения *Out* для разных значений *In1* и *In2*.

Значение *In1*	Значение *In2*	Значение *Out*
9,9	3,14	0,48
9,9	-3,14	0,48
-9,9	3,14	-0,48
-9,9	-3,14	-0,48

Ниже показан пример, в котором переменные *In1* и *In2* имеют значения REAL#-9.9 и REAL#-3.14 соответственно.
Значение переменной *abc* будет равно REAL#-0.48.

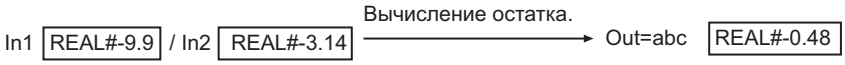
LD



ST

```
abc:=ModReal(REAL#-9.9, REAL#-3.14);
```

Команда ModReal определяет остаток от деления ***In1*** на ***In2***.
Остаток от −9.9/(−3.14) равен −0.48, поэтому ***abc*** будет равно REAL#-0.48.



Дополнительная информация

При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.

Меры предосторожности для обеспечения надлежащей эксплуатации

- В следующей таблице приведены значения *Out* для различных комбинаций значений *In1* и *In2*.

		In1				
		0	**Число**	**+ ∞**	**-∞**	**Нечисловое значение**
In2	**0**	Нечисловое значение	Нечисловое значение	Нечисловое значение	Нечисловое значение	Нечисловое значение
	Число	0	Остаток от *In1*/*In2*	Нечисловое значение	Нечисловое значение	Нечисловое значение
	+ ∞	0	Значение *In1*	Нечисловое значение	Нечисловое значение	Нечисловое значение
	-∞	0	Значение *In1*	Нечисловое значение	Нечисловое значение	Нечисловое значение
	Нечисловое значение	Нечисловое значение	Нечисловое значение	Нечисловое значение	Нечисловое значение	Нечисловое значение

- Если в *In1* или *In2* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In1* или *In2*	**Тип данных *In1* или *In2***
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL

Тип данных параметра, передаваемого в *In1* или *In2*	Тип данных *In1* или *In2*
ULINT или LINT	В этом случае произойдет ошибка сборки.

Fraction

Команда Fraction находит дробную часть вещественного числа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Fraction	Дробная часть вещественного числа	FUN	(@)Fraction EN ENO In Out	Out:=Fraction(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Вещественное число	Вход	Вещественное число	Зависит от типа данных.	---	*1
Out	Дробная часть	Выход	Дробная часть	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

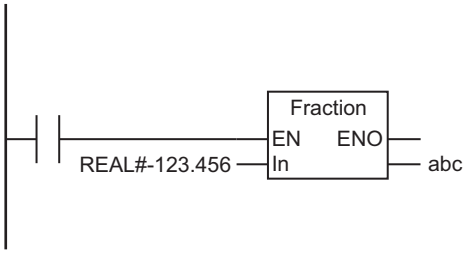
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														ОК	ОК					
Out														ОК	ОК					

Функция

Команда Fraction находит дробную часть вещественного числа *In*.

В показанном ниже примере программы *In* = REAL#-123.456.
Значение переменной *abc* будет равно REAL#-0.456.

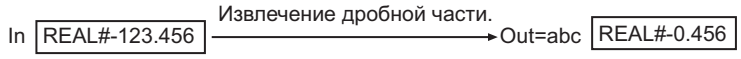
LD



ST

```
abc:=Fraction(REAL#-123.456);
```

Команда Fraction находит дробную часть ***In***.
Дробная часть −123.456 равна −0.456, поэтому ***abc*** будет равно REAL#0.456.



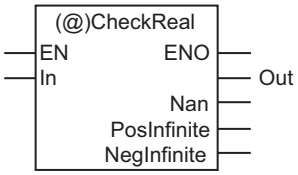
Дополнительная информация

- При выполнении вычислений над вещественными числами используйте команду *CheckReal* на стр. 2-274, чтобы проверить, не является ли *Out* положительной или отрицательной бесконечностью либо нечисловым значением.
- Если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

CheckReal

Команда CheckReal проверяет, не является ли вещественное число бесконечностью или нечисловым значением.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CheckReal	Проверка вещественного числа	FUN		CheckReal(In, Nan, PosInfinite, NegInfinite);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Вещественное число	Вход	Вещественное число	Зависит от типа данных.	---	*1
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---
Nan	Результат проверки на нечисловое значение		ИСТИНА: нечисловое значение ЛОЖЬ: не является нечисловым значением	Зависит от типа данных.		
PosInfinite	Результат проверки на положительную бесконечность		ИСТИНА: положительная бесконечность ЛОЖЬ: не является положительной бесконечностью			
NegInfinite	Результат проверки на отрицательную бесконечность		ИСТИНА: отрицательная бесконечность ЛОЖЬ: не является отрицательной бесконечностью			

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														ОК	ОК					
Out	ОК																			
Nan	ОК																			

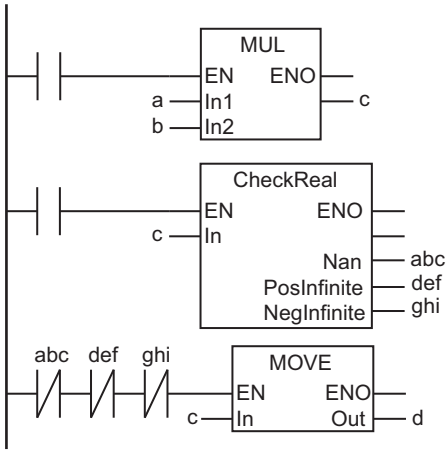
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
PosInfinite	OK																			
NegInfinite	OK																			

Функция

Команда CheckReal проверяет вещественное число *In* на предмет того, не является ли оно нечисловым значением либо положительной или отрицательной бесконечностью. Оно выводит результаты проверки в переменные *Nan*, *PosInfinite* и *NegInfinite*.

Пример программы представлен на рисунке ниже. В этом примере перемножаются переменные *a* и *b* типа REAL, после чего проверяется, является ли результат вещественным числом. Если результат умножения является вещественным числом, то он присваивается переменной *d*.

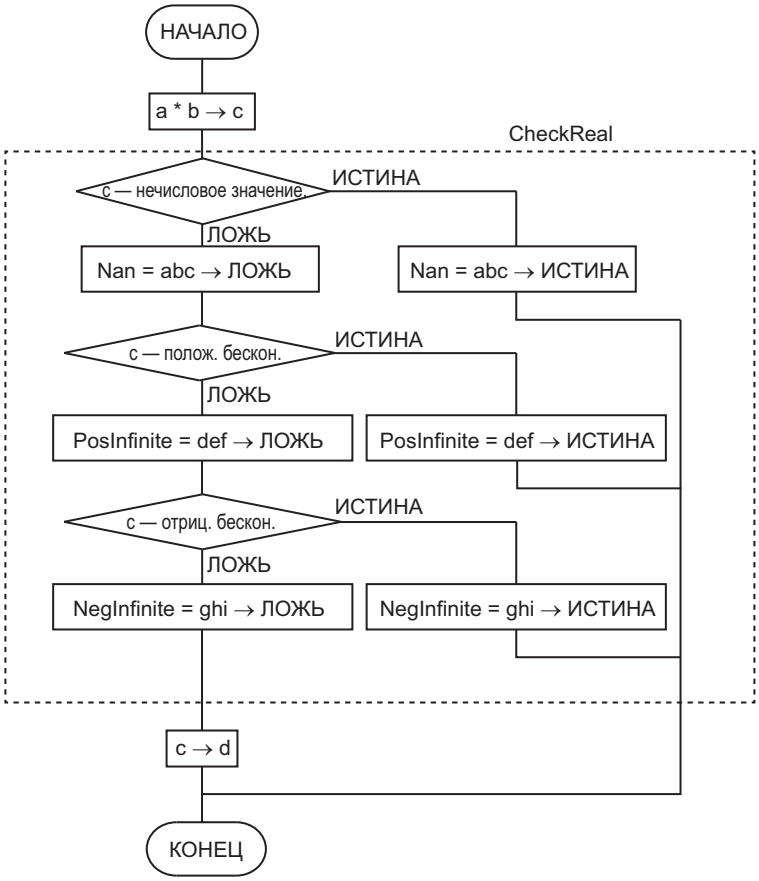
LD



ST

```
c:=a*b;
CheckReal(c, abc, def, ghi);
IF ( (abc=FALSE) AND (def=FALSE) AND (ghi=FALSE) ) THEN
    d:=c;
END_IF;
```

Если произведение ***c*** значений ***a*** и ***b*** не является нечисловым значением, а также положительной или отрицательной бесконечностью, то значение ***c*** присваивается ***d***.



Дополнительная информация

Эту команду используют при выполнении команд математических операций, оперирующих с вещественными числами. Она позволяет проверить, не является ли результат выполнения команды нечисловым значением либо положительной или отрицательной бесконечностью.

Меры предосторожности для обеспечения надлежащей эксплуатации

- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- Если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

Команды преобразования двоично-деся-тичных значений

Команда	Имя	Стр.
_BCD_TO_*	Группа преобразования BCD в целые числа без знака	стр. 2-278
_TO_BCD_*	Группа преобразования целых чисел без знака в BCD	стр. 2-281
BCD_TO_**	Группа преобразования типа данных BCD в целые числа без знака	стр. 2-284
BCDsToBin	Преобразование BCD со знаком в целое число со знаком	стр. 2-287
BinToBCDs_**	Группа преобразования целых чисел со знаком в BCD	стр. 2-290
AryToBCD	Преобразование массива в BCD	стр. 2-293
AryToBin	Преобразование массива в беззнаковые целые	стр. 2-295

_BCD_TO_*

Эти команды преобразуют битовые строки в двоично-десятичном формате (BCD) в целые числа без знака.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_BCD_TO_*	Группа преобразования BCD в целые числа без знака	FUN	(@)**_BCD_TO_*** EN ENO In Out "**" — тип данных, относящийся к битовым строкам. "***" — целочисленный тип данных.	Out:=**_BCD_TO_*** (In); "**" — тип данных, относящийся к битовым строкам. "***" — целочисленный тип данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-279 ниже.

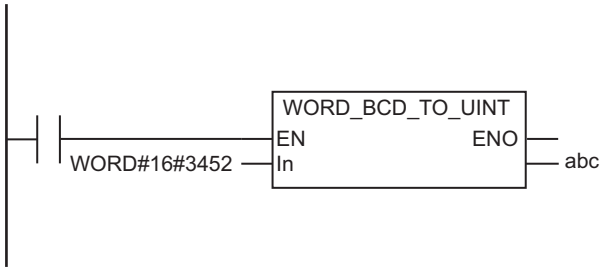
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		ОК	ОК	ОК	ОК															
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК							

Функция

Эти команды преобразуют значение *In* (данные для преобразования) (которое должно быть битовой строкой в двоично-десятичном формате (BCD)) в целочисленное значение без знака. Имя команды определяется типами данных *In* и *Out* (результат преобразования). Например, если *In* относится к типу данных WORD, а *Out* относится к типу данных UINT, команда будет иметь имя WORD_BCD_TO_UINT.

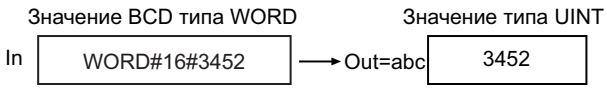
Ниже показан пример для команды WORD_BCD_TO_UINT, в котором *In* = WORD16#3452.

LD



ST

```
abc:=WORD_BCD_TO_UINT(WORD#16#3452);
```



Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
BYTE	USINT	16#00...16#99 (BCD)	0...99
	UINT		
	UDINT		
	ULINT		
	SINT		
	INT		
	DINT		
	LINT		
WORD	USINT	16#0000...16#0255 (BCD)	0...255
	UINT	16#0000...16#9999 (BCD)	0...9999
	UDINT		
	ULINT		
	SINT	16#0000...16#0127 (BCD)	0...127
	INT	16#0000...16#9999 (BCD)	0...9999
	DINT		
	LINT		
DWORD	USINT	16#0000_0000...16#0000_0255 (BCD)	0...255
	UINT	16#0000_0000...16#0006_5535 (BCD)	0...65535
	UDINT	16#0000_0000...16#9999_9999 (BCD)	0...99999999
	ULINT		
	SINT	16#0000_0000...16#0000_0127 (BCD)	0...127
	INT	16#0000_0000...16#0003_2767 (BCD)	0...32767
	DINT	16#0000_0000...16#9999_9999 (BCD)	0...99999999
	LINT		

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
LWORD	USINT	16#0000_0000_0000_0000...16#0000_0000_0000_0255 (BCD)	0...255
	UINT	16#0000_0000_0000_0000...16#0000_0000_0006_5535 (BCD)	0...65535
	UDINT	16#0000_0000_0000_0000...16#0000_0042_9496_7295 (BCD)	0...4294967295
	ULINT	16#0000_0000_0000_0000...16#9999_9999_9999_9999 (BCD)	0...9999999999999999
	SINT	16#0000_0000_0000_0000...16#0000_0000_0000_0127 (BCD)	0...127
	INT	16#0000_0000_0000_0000...16#0000_0000_0003_2767 (BCD)	0...32767
	DINT	16#0000_0000_0000_0000...16#0000_0021_4748_3647 (BCD)	0...2147483647
	LINT	16#0000_0000_0000_0000...16#9999_9999_9999_9999 (BCD)	0...9999999999999999

Дополнительная информация

- Для преобразования битовой строки в формате BCD в целое число используйте команду *BCD_TO_*** на стр. 2-284.
- Для преобразования целого числа в битовую строку в формате BCD используйте команду ***_TO_BCD_**** на стр. 2-281.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.
- Если размер (разрядность) значений *Out* превышает размер (разрядность) значений *In*, старшие разряды *Out* будут содержать 0.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *In* находится за пределами допустимого диапазона.
  b) Значение в *In* не является битовой строкой в формате BCD (т. е. содержит шестнадцатеричное значение A, B, C, D, E или F).

_TO_BCD_*

Эти команды преобразуют целые числа без знака в битовые строки в двоично-десятичном формате (BCD).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_TO_BCD_*	Группа преобразования целых чисел без знака в BCD	FUN	(@)**_TO_BCD_*** EN ENO In Out "**" — целочисленный тип данных. "****" — тип данных, относящийся к битовым строкам.	Out:=**_TO_BCD_*** (In); "**" — целочисленный тип данных. "****" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-282 ниже.

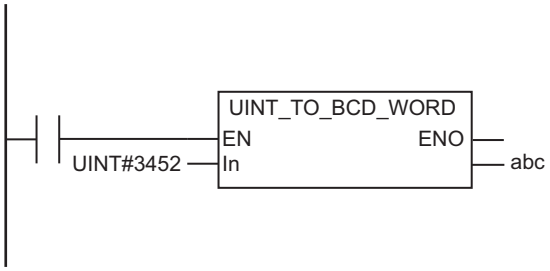
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						OK	OK	OK	OK	OK	OK	OK	OK							
Out		OK	OK	OK	OK															

Функция

Эти команды преобразуют значение *In* (данные для преобразования) (которое должно быть целочисленным значением без знака) в битовую строку в двоично-десятичном формате (BCD). Имя команды определяется типами данных *In* и *Out* (результат преобразования). Например, если *In* относится к типу данных UINT, а *Out* относится к типу данных WORD, команда будет иметь имя UINT_TO_BCD_WORD.

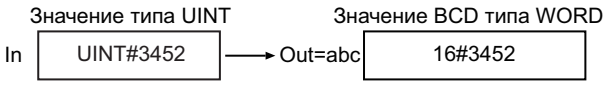
Ниже показан пример для команды UINT_TO_BCD_WORD, в котором *In* = UNIT#3452.

LD



ST

```
abc:=UINT_TO_BCD_WORD(UINT#3452);
```



Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
USINT	BYTE	0...99	16#00...16#99 (BCD)
	WORD	0...255	16#0000...16#0255 (BCD)
	DWORD		16#0000_0000...16#0000_0255 (BCD)
	LWORD		16#0000_0000_0000_0000...16#0000_0000_0000_0255 (BCD)
UINT	BYTE	0...99	16#00...16#99 (BCD)
	WORD	0...9999	16#0000...16#9999 (BCD)
	DWORD	0...65535	16#0000_0000...16#0006_5535 (BCD)
	LWORD		16#0000_0000_0000_0000...16#0000_0000_0006_5535 (BCD)
UDINT	BYTE	0...99	16#00...16#99 (BCD)
	WORD	0...9999	16#0000...16#9999 (BCD)
	DWORD	0...99999999	16#0000_0000...16#9999_9999 (BCD)
	LWORD	0...4294967295	16#0000_0000_0000_0000...16#0000_0042_9496_7295 (BCD)
ULINT	BYTE	0...99	16#00...16#99 (BCD)
	WORD	0...9999	16#0000...16#9999 (BCD)
	DWORD	0...99999999	16#0000_0000...16#9999_9999 (BCD)
	LWORD	0...9999999999999999	16#0000_0000_0000_0000...16#9999_9999_9999_9999 (BCD)
SINT	BYTE	0...99	16#00...16#99 (BCD)
	WORD	0...127	16#0000...16#0127 (BCD)
	DWORD		16#0000_0000...16#0000_0127 (BCD)
	LWORD		16#0000_0000_0000_0000...16#0000_0000_0000_0127 (BCD)
INT	BYTE	0...99	16#00...16#99 (BCD)
	WORD	0...9999	16#0000...16#9999 (BCD)
	DWORD	0...32767	16#0000_0000...16#0003_2767 (BCD)
	LWORD		16#0000_0000_0000_0000...16#0000_0000_0003_2767 (BCD)

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
DINT	BYTE	0...99	16#00...16#99 (BCD)
	WORD	0...9999	16#0000...16#9999 (BCD)
	DWORD	0...99999999	16#0000_0000...16#9999_9999 (BCD)
	LWORD	0...2147483647	16#0000_0000_0000_0000...16#0000_0021_4748_3647 (BCD)
LINT	BYTE	0...99	16#00...16#99 (BCD)
	WORD	0...9999	16#0000...16#9999 (BCD)
	DWORD	0...99999999	16#0000_0000...16#9999_9999 (BCD)
	LWORD	0...9999999999999999	16#0000_0000_0000_0000...16#9999_9999_9999_9999 (BCD)

Дополнительная информация

- Для преобразования битовой строки в формате BCD конкретного типа в целое число используйте команду ***_BCD_TO_*** на стр. 2-278.
- Для преобразования битовой строки в формате BCD в целое число используйте команду *BCD_TO_*** на стр. 2-284.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.
- Если размер (разрядность) значений *Out* превышает размер (разрядность) значений *In*, старшие разряды *Out* будут содержать 0.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *In* находится за пределами допустимого диапазона.

BCD_TO_**

Команда BCD_TO_** преобразует битовые строки в двоично-десятичном формате (BCD) в целые числа без знака.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
BCD_TO_**	Группа преобразования типа данных BCD в целые числа без знака	FUN	(@)BCD_TO_** EN ENO In Out "**" — целочисленный тип данных.	Out:=BCD_TO_** (In); "**" — целочисленный тип данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	*2
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-285 ниже.

*2. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

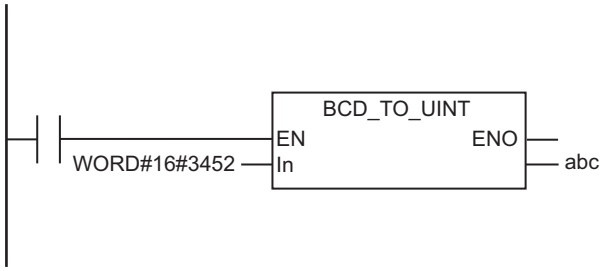
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Out						OK	OK	OK	OK	OK	OK	OK	OK							

Функция

Эти команды преобразуют значение *In* (данные для преобразования) (которое должно быть битовой строкой в двоично-десятичном формате (BCD)) в целочисленное значение без знака. Имя команды определяется типом данных *Out* (результат преобразования). Например, если *Out* относится к типу данных UINT, команда будет иметь имя BCD_TO_UINT.

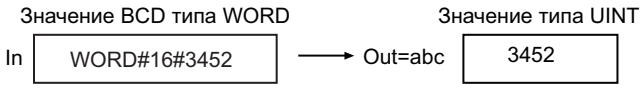
Ниже показан пример для команды BCD_TO_UINT, в котором *In* = WORD#16#3452.

LD



ST

```
abc:=BCD_TO_UINT(WORD#16#3452);
```



Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
BYTE	USINT	16#00...16#99 (BCD)	0...99
	UINT		
	UDINT		
	ULINT		
	SINT		
	INT		
	DINT		
	LINT		
WORD	USINT	16#0000...16#0255 (BCD)	0...255
	UINT	16#0000...16#9999 (BCD)	0...9999
	UDINT		
	ULINT		
	SINT	16#0000...16#0127 (BCD)	0...127
	INT	16#0000...16#9999 (BCD)	0...9999
	DINT		
	LINT		
DWORD	USINT	16#0000_0000...16#0000_0255 (BCD)	0...255
	UINT	16#0000_0000...16#0006_5535 (BCD)	0...65535
	UDINT	16#0000_0000...16#9999_9999 (BCD)	0...99999999
	ULINT		
	SINT	16#0000_0000...16#0000_0127 (BCD)	0...127
	INT	16#0000_0000...16#0003_2767 (BCD)	0...32767
	DINT	16#0000_0000...16#9999_9999 (BCD)	0...99999999
	LINT		

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
LWORD	USINT	16#0000_0000_0000_0000...16#0000_0000_0000_0255 (BCD)	0...255
	UINT	16#0000_0000_0000_0000...16#0000_0000_0006_5535 (BCD)	0...65535
	UDINT	16#0000_0000_0000_0000...16#0000_0042_9496_7295 (BCD)	0...4294967295
	ULINT	16#0000_0000_0000_0000...16#9999_9999_9999_9999 (BCD)	0...9999999999999999
	SINT	16#0000_0000_0000_0000...16#0000_0000_0000_0127 (BCD)	0...127
	INT	16#0000_0000_0000_0000...16#0000_0000_0003_2767 (BCD)	0...32767
	DINT	16#0000_0000_0000_0000...16#0000_0021_4748_3647 (BCD)	0...2147483647
	LINT	16#0000_0000_0000_0000...16#9999_9999_9999_9999 (BCD)	0...9999999999999999

Дополнительная информация

- Для преобразования битовой строки в формате BCD конкретного типа в целое число используйте команду ***_BCD_TO_**** на стр. 2-278.
- Для преобразования целого числа в битовую строку в формате BCD используйте команду ***_TO_BCD_**** на стр. 2-281.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типу данных переменной *Out*.
- Если размер (разрядность) значений *Out* превышает размер (разрядность) значений *In*, старшие разряды *Out* будут содержать 0.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *In* находится за пределами допустимого диапазона.
  b) Значение в *In* не является битовой строкой в формате BCD (т. е. содержит шестнадцатеричное значение A, B, C, D, E или F).

BCDsToBin

Команда BCDsToBin преобразует битовые строки в двоично-десятичном формате (BCD) со знаком в целые числа со знаком.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
BCDsToBin	Преобразование BCD со знаком в целое число со знаком	FUN	(@)BCDsToBin EN ENO In Out Format	Out:=BCDsToBin(In, Format);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	*2
Format	Номер формата представления		Формат представления битовой строки в двоично-десятичном коде (BCD)	_BCD0..._BCD3		_BCD0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Допустимый диапазон зависит от значения *Format*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-288 ниже.

*2. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Format	Сведения о перечислителях перечислимого типа _eBCD_FORMAT см. в разделе *Функция* на стр. 2-287.																			
Out	Должен использоваться целочисленный тип данных со знаком того же размера, что и *In*.																			

Функция

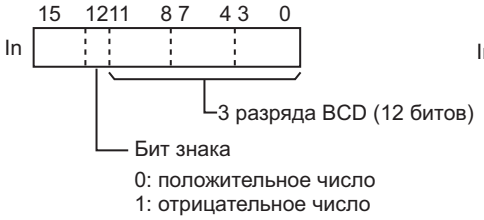
Команда BCDsToBin преобразует битовую строку в двоично-десятичном формате (BCD) со знаком *In* в целое число со знаком.

Для значения номера формата представления *Format* используется перечислимый тип данных _eBCD_FORMAT.

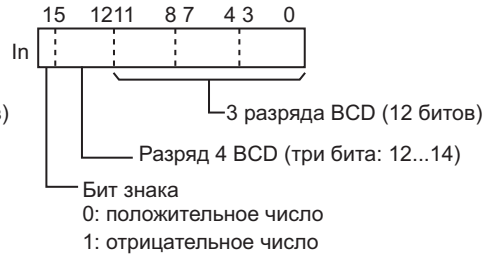
Выберите одно из следующих значений: _BCD0, _BCD1, _BCD2 или _BCD3. Указание знака в четырех старших битах *In* зависит от номера формата BCD.

Ниже показаны примеры форматов представления данных для значения *In* типа WORD.

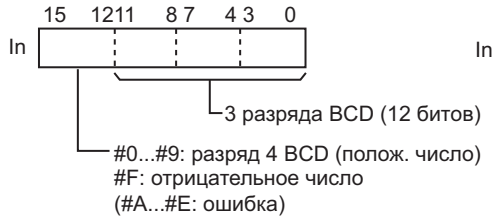
Формат = _BCD0
Допустимый диапазон ***In***: -999...999 (BCD)



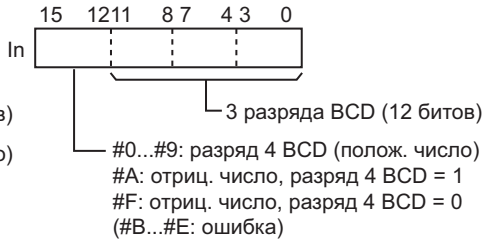
Формат = _BCD1
Допустимый диапазон ***In***: -7999...7999 (BCD)



Формат = _BCD2
Допустимый диапазон ***In***: -999...9999 (BCD)



Формат = _BCD3
Допустимый диапазон ***In***: -1999...9999 (BCD)



Допустимый диапазон

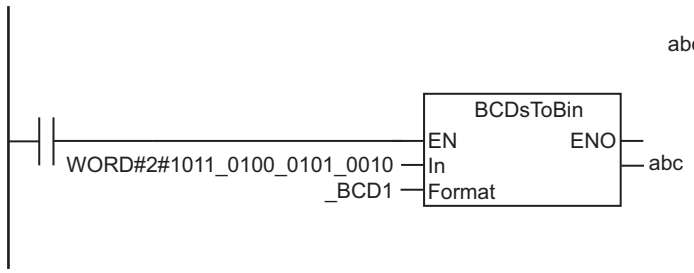
Типы данных *In* и *Out* должны быть одного размера. Допустимые диапазоны зависят от значения *Format*, как показано ниже.

		Значение *Format*			
		_BCD0	**_BCD1**	**_BCD2**	**_BCD3**
Тип данных *In* ↓ Тип данных *Out*	**BYTE ↓ SINT**	-9...9	-79...79	-9...99	-19...99
	WORD ↓ INT	-999...999	-7999...7999	-999...9999	-1999...9999
	DWORD ↓ DINT	-9999999...9999999	-79999999...79999999	-9999999...99999999	-19999999...99999999
	LWORD ↓ LINT	-999999999999999...999999999999999	-7999999999999999...7999999999999999	-999999999999999...999999999999999	-1999999999999999...9999999999999999

Пример записи

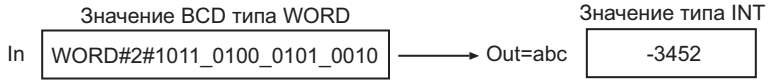
Ниже приведен пример, в котором *In* = WORD#2#1011_0100_0101_0010, а *Format* = _BCD1.

LD



ST

```
abc:=BCDsToBin(WORD#2#1011_0100_0101_0010,
                _BCD1);
```

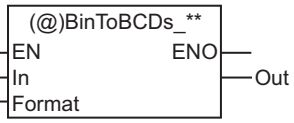


Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных *In* и *Out* должны быть одного размера.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Format* = _BCD0, а старший разряд *In* = 2...F.
  b) Значение *Format* = _BCD2, а старший разряд *In* = A...E.
  c) Значение *Format* = _BCD3, а старший разряд *In* = B...E.
  d) Какой-либо разряд в *In* = A...F, за исключением вышеуказанных условий.
  e) Значение *Format* находится за пределами допустимого диапазона.

BinToBCDs_**

Эти команды преобразуют целые числа со знаком в битовые строки в двоично-десятичном формате (BCD) со знаком.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
BinToBCDs_**	Группа преобразования целых чисел со знаком в BCD	FUN	 "**" — тип данных, относящийся к битовым строкам.	Out:=BinToBCDs(In, Format); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Format	Номер формата представления		Формат представления битовой строки в двоично-десятичном коде (BCD)	_BCD0..._BCD3		_BCD0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Допустимый диапазон зависит от значения *Format*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-291 ниже.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In										OK	OK	OK	OK							
Format	Сведения о перечислителях перечислимого типа _eBCD_FORMAT см. в разделе *Функция* на стр. 2-290.																			
Out	Тип данных должен быть таким же, как у *In*.																			

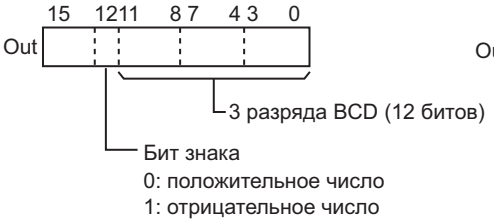
Функция

Эти команды преобразуют целое число со знаком *In* в битовую строку в двоично-десятичном формате (BCD) со знаком.
Имя команды определяется типом данных *Out*. Например, если *Out* относится к типу данных WORD, команда будет иметь имя BinToBCDs_WORD.

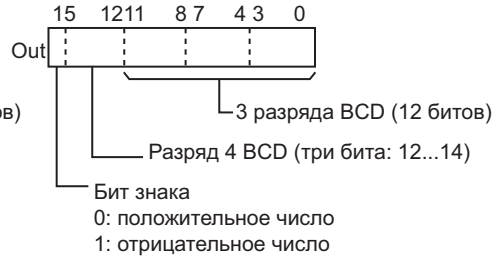
Для значения номера формата представления *Format* используется перечислимый тип данных _eBCD_FORMAT.
Выберите одно из следующих значений: _BCD0, _BCD1, _BCD2 или _BCD3. Указание знака в четырех старших битах *Out* зависит от номера формата BCD.

Ниже показаны примеры форматов представления данных для значения *Out* типа WORD.

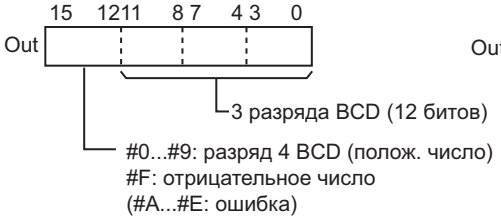
Формат = _BCD0
Допустимый диапазон ***Out***: -999...999



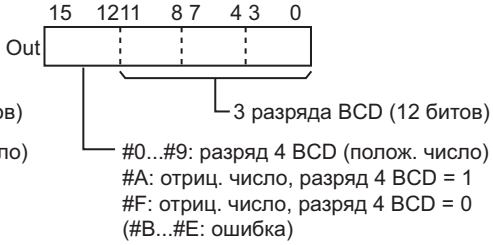
Формат = _BCD1
Допустимый диапазон ***Out***: -7999...7999



Формат = _BCD2
Допустимый диапазон ***Out***: -999...9999 (BCD)



Формат = _BCD3
Допустимый диапазон ***Out***: -1999...9999



Допустимый диапазон

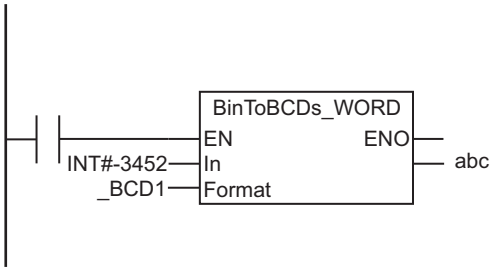
Типы данных *In* и *Out* должны быть одного размера. Допустимые диапазоны зависят от значения *Format*, как показано ниже.

		Значение *Format*			
		_BCD0	**_BCD1**	**_BCD2**	**_BCD3**
Тип данных *In* ↓ Тип данных *Out*	**SINT ↓ BYTE**	-9...9	-79...79	-9...99	-19...99
	INT ↓ WORD	-999...999	-7999...7999	-999...9999	-1999...9999
	DINT ↓ DWORD	-9999999...9999999	-79999999...79999999	-9999999...99999999	-19999999...99999999
	LINT ↓ LWORD	-999999999999999...9999999999999999	-7999999999999999...7999999999999999	-999999999999999...9999999999999999	-1999999999999999...9999999999999999

Пример записи

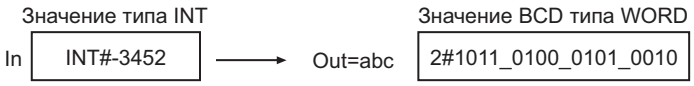
Ниже приведен пример использования команды BinToBCDs_WORD для случая, когда *In* = INT#-3452, а *Format* = _BCD1.

LD



ST

```
abc:=BinToBCDs_WORD(INT#-3452, _BCD1);
```

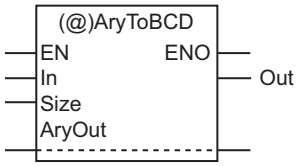


Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типу данных переменной *Out*.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *In* находится за пределами допустимого диапазона.
  b) Значение *Format* находится за пределами допустимого диапазона.

AryToBCD

Команда AryToBCD преобразует элементы массива, являющиеся целыми числами без знака, в битовые строки в двоично-десятичном формате (BCD).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryToBCD	Преобразование массива в BCD	FUN		AryToBCD(In, Size, AryOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив целых чисел без знака	Вход	Массив целых чисел без знака	*1	---	*2
Size	Количество элементов		Количество элементов In[] для преобразования	Зависит от типа данных.		1
AryOut[] (массив)	Массив BCD	Вход-выход	Массив BCD	*1	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Диапазоны допустимых значений зависят от типов данных элементов массивов In[] и AryOut[]. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-294.

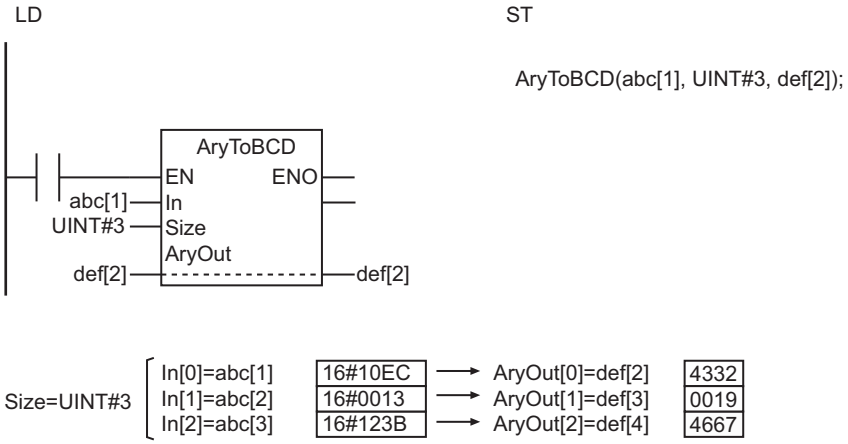
*2. Если входной параметр будет опущен, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)						ОК	ОК	ОК	ОК											
Size							ОК													
AryOut[] (массив)	Должен быть массивом битовых строк. Тип данных должен быть того же размера, что и элементы In[].																			
Out	ОК																			

Функция

Команда AryToBCD преобразует *Size* элементов массива целочисленных значений без знака In[], начиная с элемента In[0], в битовые строки в двоично-десятичном формате (BCD). Битовые строки в формате BCD выводятся в массив AryOut[].

Ниже приведен пример, в котором *Size* = UINT#3.



Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для массивов In[] и AryOut[] в соответствии с типами данных их элементов.

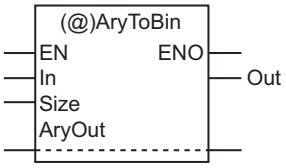
Тип данных элементов In[]	Тип данных элементов AryOut[]	Допустимый диапазон для In[]	Допустимый диапазон для AryOut[]
USINT	BYTE	0...99	16#00...16#99 (BCD)
UINT	WORD	0...9999	16#0000...16#9999 (BCD)
UDINT	DWORD	0...99999999	16#0000_0000...16#9999_9999 (BCD)
ULINT	LWORD	0...9999999999999999	16#0000_0000_0000_0000...16#9999_9999_9999_9999 (BCD)

Меры предосторожности для обеспечения надлежащей эксплуатации

- Для массивов In[] и AryOut[] следует использовать типы данных одинакового размера. Например, если элементы массива In[] являются значениями типа UINT, для элементов массива AryOut[] следует использовать тип данных WORD. В противном случае произойдет ошибка сборки.
- Эта команда не преобразует двоичное значение со знаком в значение в формате BCD со знаком. Для массива In[] следует использовать целочисленный тип данных без знака (USINT, UINT, UDINT или ULINT).
- Если *Size* = 0, значения в массиве AryOut[] не изменяются.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение In[] находится за пределами допустимого диапазона.
  b) Значение *Size* приводит к выходу за область массива In[] или AryOut[].

AryToBin

Команда AryToBin преобразует элементы массива, являющиеся битовыми строками в двоично-десятичном формате (BCD), в целые числа без знака.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryToBin	Преобразование массива в беззнаковые целые	FUN		AryToBin(In, Size, AryOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив битовых строк в формате BCD	Вход	Массив битовых строк в формате BCD	*1	---	*2
Size	Количество элементов		Количество элементов In[] для преобразования	Зависит от типа данных.		1
AryOut[] (массив)	Массив целых чисел без знака	Вход-выход	Массив целых чисел без знака	*1	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Диапазоны допустимых значений зависят от типов данных элементов массивов In[] и AryOut[]. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-296.

*2. Если входной параметр будет опущен, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

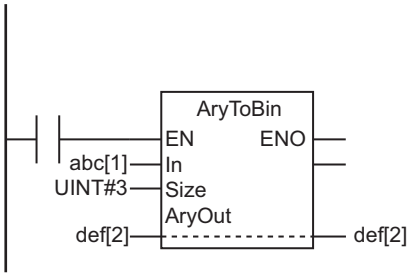
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)		OK	OK	OK	OK															
Size							OK													
AryOut[] (массив)	Должен быть массивом целых чисел без знака Тип данных должен быть того же размера, что и элементы In[].																			
Out	OK																			

Функция

Команда AryToBin преобразует *Size* элементов массива битовых строк в двоично-десятичном формате (BCD), начиная с In[0], в целочисленные значения без знака. Целочисленные значения без знака выводятся в массив AryOut[].

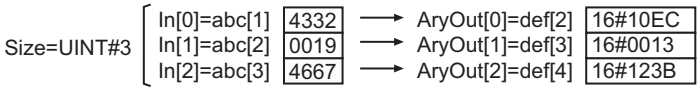
Ниже приведен пример, в котором *Size* = UINT#3.

LD



ST

```
AryToBin(abc[1], UINT#3, def[2]);
```



Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для массивов In[] и AryOut[] в соответствии с типами данных их элементов.

Тип данных элементов In[]	Тип данных элементов AryOut[]	Допустимый диапазон для In[]	Допустимый диапазон для AryOut[]
BYTE	USINT	16#00...16#99 (BCD)	0...99
WORD	UINT	16#0000...16#9999 (BCD)	0...9999
DWORD	UDINT	16#0000_0000...16#9999_9999 (BCD)	0...99999999
LWORD	ULINT	16#0000_0000_0000_0000...16#9999_9999_9999_9999 (BCD)	0...9999999999999999

Меры предосторожности для обеспечения надлежащей эксплуатации

- Для массивов In[] и AryOut[] следует использовать типы данных одинакового размера. Например, если элементы массива In[] являются значениями типа WORD, для элементов массива AryOut[] следует использовать тип данных USINT. В противном случае произойдет ошибка сборки.
- Эта команда не преобразует значение в формате BCD со знаком в двоичное значение со знаком. Для массива AryOut [] следует использовать целочисленный тип данных без знака (USINT, UINT, UDINT или ULINT).
- Если *Size* = 0, значения в массиве AryOut[] не изменяются.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение *Size* приводит к выходу за область массива In[] или AryOut[].
  b) Значение в In[] не является битовой строкой в формате BCD (т. е. содержит шестнадцатеричное значение A, B, C, D, E или F).

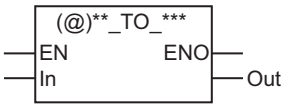
Команды преобразования типов данных

Команда	Имя	Стр.
_TO_* (Группа преобразования целых чисел в целые числа)	Группа преобразования целых чисел в целые числа	стр. 2-299
_TO_* (Группа преобразования целых чисел в битовые строки)	Группа преобразования целых чисел в битовые строки	стр. 2-302
_TO_* (Группа преобразования целых чисел в вещественные числа)	Группа преобразования целых чисел в вещественные числа	стр. 2-305
_TO_* (Группа преобразования битовых строк в целые числа)	Группа преобразования битовых строк в целые числа	стр. 2-308
_TO_* (Группа преобразования битовых строк в битовые строки)	Группа преобразования битовых строк в битовые строки	стр. 2-311
_TO_* (Группа преобразования битовых строк в вещественные числа)	Группа преобразования битовых строк в вещественные числа	стр. 2-313
_TO_* (Группа преобразования вещественных чисел в целые числа)	Группа преобразования вещественных чисел в целые числа	стр. 2-315
_TO_* (Группа преобразования вещественных чисел в битовые строки)	Группа преобразования вещественных чисел в битовые строки	стр. 2-318
_TO_* (Группа преобразования вещественных чисел в вещественные числа)	Группа преобразования вещественных чисел в вещественные числа	стр. 2-321
**_TO_STRING (Группа преобразования целых чисел в текстовые строки)	Группа преобразования целых чисел в текстовые строки	стр. 2-323
**_TO_STRING (Группа преобразования битовых строк в текстовые строки)	Группа преобразования битовых строк в текстовые строки	стр. 2-325
**_TO_STRING (Группа преобразования вещественных чисел в текстовые строки)	Группа преобразования вещественных чисел в текстовые строки	стр. 2-327
RealToFormatString	REAL в форматированную текстовую строку	стр. 2-330

Команда	Имя	Стр.
LrealToFormatString	LREAL в форматированную текстовую строку	стр. 2-337
STRING_TO_** (Группа преобразования текстовых строк в целые числа)	Группа преобразования текстовых строк в целые числа	стр. 2-344
STRING_TO_** (Группа преобразования текстовых строк в битовые строки)	Группа преобразования текстовых строк в битовые строки	стр. 2-347
STRING_TO_** (Группа преобразования текстовых строк в вещественные числа)	Группа преобразования текстовых строк в вещественные числа	стр. 2-350
TO_** (Группа преобразования в целые числа)	Группа преобразования в целые числа	стр. 2-354
TO_** (Группа преобразования в битовые строки)	Группа преобразования в битовые строки	стр. 2-357
TO_** (Группа преобразования в вещественные числа)	Группа преобразования в вещественные числа	стр. 2-359
EnumToNum	Перечисление в целое число	стр. 2-361
NumToEnum	Целое число в перечисление	стр. 2-363
TRUNC, Round и RoundUp	Усечение/Округление вещественного числа/ Округление вещественного числа к большему	стр. 2-366

_TO_* (Группа преобразования целых чисел в целые числа)

Эти команды служат для преобразования целочисленных значений одного типа в целочисленные значения другого типа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_TO_*	Группа преобразования целых чисел в целые числа	FUN	 "**" и "***" — отличающиеся целочисленные типы данных.	Out:=**_TO_*** (In); "**" и "***" — отличающиеся целочисленные типы данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-300 ниже.

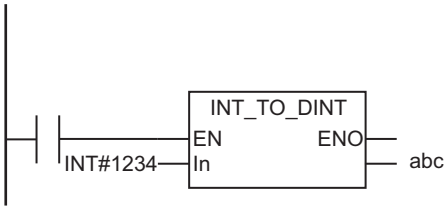
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						OK	OK	OK	OK	OK	OK	OK	OK							
Out						OK	OK	OK	OK	OK	OK	OK	OK							

Функция

Эти команды преобразуют целочисленное значение *In* одного типа в целочисленное значение другого типа.
Имя команды определяется типами данных *In* и *Out* (результат преобразования). Например, если *In* относится к типу данных INT, а *Out* относится к типу данных DINT, команда будет иметь имя INT_TO_DINT.

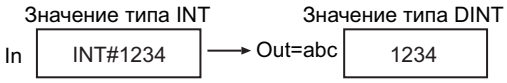
Ниже показан пример для команды INT_TO_DINT, в котором *In* = INT#1234.

LD



ST

```
abc:=INT_TO_DINT(INT#1234);
```



Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In* и *Out*
USINT	UINT	0...255
	UDINT	
	ULINT	
	SINT	0...127
	INT	0...255
	DINT	
	LINT	
UINT	USINT	0...255
	UDINT	0...65535
	ULINT	
	SINT	0...127
	INT	0...32767
	DINT	0...65535
	LINT	
UDINT	USINT	0...255
	UINT	0...65535
	ULINT	0...4294967295
	SINT	0...127
	INT	0...32767
	DINT	0...2147483647
	LINT	0...4294967295
ULINT	USINT	0...255
	UINT	0...65535
	UDINT	0...4294967295
	SINT	0...127
	INT	0...32767
	DINT	0...2147483647
	LINT	0...9223372036854775807

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In* и *Out*
SINT	USINT	0...127
	UINT	
	UDINT	
	ULINT	
	INT	-128...127
	DINT	
	LINT	
INT	USINT	0...255
	UINT	0...32767
	UDINT	
	ULINT	
	SINT	-128...127
	DINT	-32768...32767
	LINT	
DINT	USINT	0...255
	UINT	0...65535
	UDINT	0...2147483647
	ULINT	
	SINT	-128...127
	INT	-32768...32767
	LINT	-2147483648...2147483647
LINT	USINT	0...255
	UINT	0...65535
	UDINT	0...4294967295
	ULINT	0...9223372036854775807
	SINT	-128...127
	INT	-32768...32767
	DINT	-2147483648...2147483647

Дополнительная информация

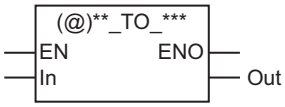
Для преобразования значения любого типа данных в целочисленное значение используйте команду *TO_** (Группа преобразования в целые числа)* на стр. 2-354.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.
- Если *In* является целым числом со знаком и разрядность *Out* больше разрядности *In*, то выполняется расширение знака.
- Если *In* является целым числом без знака и разрядность *Out* больше разрядности *In*, то старшие разряды *Out* будут содержать 0.
- Если разрядность *Out* меньше разрядности *In*, то старшие разряды будут отброшены.

_TO_* (Группа преобразования целых чисел в битовые строки)

Эти команды преобразуют целочисленные значения в битовые строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_TO_*	Группа преобразования целых чисел в битовые строки	FUN	 "**" — целочисленный тип данных. "***" — тип данных, относящийся к битовым строкам.	Out:=**_TO_*** (In); "**" — целочисленный тип данных. "***" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-303 ниже.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						OK	OK	OK	OK	OK	OK	OK	OK *1							
Out		OK	OK *1	OK	OK															

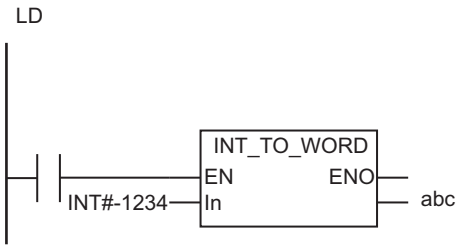
*1. В случае модуля ЦПУ NX1P2 для использования команды LINT_TO_WORD требуются версия модуля 1.14 или более поздняя и Sysmac Studio версии 1.18 или выше.

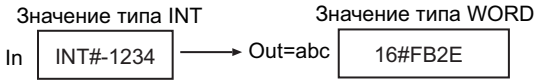
Функция

Эти команды преобразуют целочисленное значение *In* в битовую строку.

Имя команды определяется типами данных *In* и *Out* (результат преобразования). Например, если *In* относится к типу данных INT, а *Out* относится к типу данных WORD, команда будет иметь имя INT_TO_WORD.

Ниже показан пример для команды INT_TO_WORD, в котором *In* = INT#-1234.





Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
USINT	BYTE	0...255	16#00...16#FF
	WORD		
	DWORD		
	LWORD		
UINT	BYTE	0...255	16#00...16#FF
	WORD	0...65535	16#0000...16#FFFF
	DWORD		
	LWORD		
UDINT	BYTE	0...255	16#00...16#FF
	WORD	0...65535	16#0000...16#FFFF
	DWORD	0...4294967295	16#0000_0000...16#FFFF_FFFF
	LWORD		
ULINT	BYTE	0...255	16#00...16#FF
	WORD	0...65535	16#0000...16#FFFF
	DWORD	0...4294967295	16#0000_0000...16#FFFF_FFFF
	LWORD	0...18446744073709551645	16#0000_0000_0000_0000...16#FFFF_FFFF_FFFF_FFFF
SINT	BYTE	-128...127	16#00...16#FF
	WORD		
	DWORD		
	LWORD		
INT	BYTE	-128...127	16#00...16#FF
	WORD	-32768...32767	16#0000...16#FFFF
	DWORD		
	LWORD		

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
DINT	BYTE	-128...127	16#00...16#FF
	WORD	-32768...32767	16#0000...16#FFFF
	DWORD	-2147483648...2147483647	16#0000_0000...16#FFFF_FFFF
	LWORD		
LINT	BYTE	-128...127	16#00...16#FF
	WORD	-32768...32767	16#0000...16#FFFF
	DWORD	-2147483648...2147483647	16#0000_0000...16#FFFF_FFFF
	LWORD	-9223372036854775808...9223372036854775807	16#0000_0000_0000_0000...16#FFFF_FFFF_FFFF_FFFF

Дополнительная информация

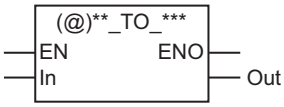
- Для преобразования битовой строки в целое число используйте команду ***_TO_*** (Группа преобразования битовых строк в целые числа)* на стр. 2-308.
- Для преобразования значения любого типа данных в битовую строку используйте команду *TO_** (Группа преобразования в битовые строки)* на стр. 2-357.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.
- Если *In* является целым числом со знаком и разрядность *Out* больше разрядности *In*, то выполняется расширение знака.
- Если *In* является целым числом без знака и разрядность *Out* больше разрядности *In*, то старшие разряды *Out* будут содержать 0.
- Если разрядность *Out* меньше разрядности *In*, то старшие разряды будут отброшены.

_TO_* (Группа преобразования целых чисел в вещественные числа)

Эти команды преобразуют целочисленные значения в вещественные значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_TO_*	Группа преобразования целых чисел в вещественные числа	FUN	 "**" — целочисленный тип данных. "***" — вещественный тип данных.	Out:=**_TO_*** (In); "**" — целочисленный тип данных. "***" — вещественный тип данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-306 ниже.

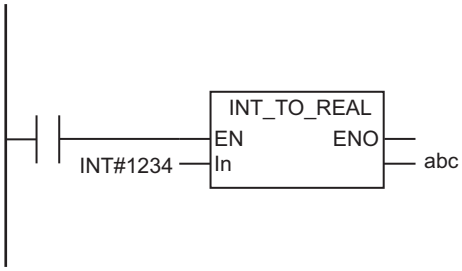
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК							
Out														ОК	ОК					

Функция

Эти команды преобразуют целочисленное значение *In* в вещественное значение.
Имя команды определяется типами данных *In* и *Out* (результат преобразования). Например, если *In* относится к типу данных INT, а *Out* относится к типу данных REAL, команда будет иметь имя INT_TO_REAL.

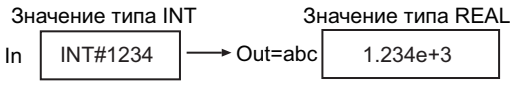
Ниже показан пример для команды INT_TO_REAL, в котором *In* = INT#1234.

LD



ST

```
abc:=INT_TO_REAL(INT#1234);
```



Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
USINT	REAL	0...255	0...2,55e+2
	LREAL		
UINT	REAL	0...65535	0...6,5535e+4
	LREAL		
UDINT	REAL	0...4294967295	0...4,294967e+9
	LREAL		0...4,294967295e+9
ULINT	REAL	0...18446744073709551615	0...1,844674e+19
	LREAL		0...1,84467440737095e+19
SINT	REAL	-128...127	-1,28e+2...1,27e+2
	LREAL		
INT	REAL	-32768...32767	-3,2768e+4...3,2767e+4
	LREAL		
DINT	REAL	-2147483648...2147483647	-2,147483e+9...2,147483e+9
	LREAL		-2,147483648e+9...2,147483647e+9
LINT	REAL	-9223372036854775808...9223372036854775807	-9,223372e+18...9,223372e+18
	LREAL		-9,22337203685477e+18...9,22337203685477e+18

Дополнительная информация

- Для преобразования вещественного числа в целое число используйте команду ***_TO_*** (Группа преобразования вещественных чисел в целые числа)* на стр. 2-315.
- Для преобразования значения любого типа данных в вещественное число используйте команду *TO_** (Группа преобразования в вещественные числа)* на стр. 2-359.

Меры предосторожности для обеспечения надлежащей эксплуатации

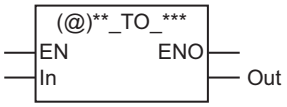
- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.

- В значащих разрядах выходного вещественного числа производится округление в зависимости от типов данных переменных *In* и *Out*. Из-за этого выходное значение может не быть точно равно входному значению. В следующей таблице перечислены комбинации типов данных, приводящие к возникновению ошибки.

Тип данных *In*	Тип данных *Out*	Значения, при которых возникает ошибка
DINT	REAL	-16777216 или меньше либо 16777216 или больше
LINT		
UDINT	REAL	16777216 или больше
ULINT		
LINT	LREAL	-9007199254740992 или меньше либо 9007199254740992 или больше
ULINT	LREAL	9007199254740992 или больше

_TO_* (Группа преобразования битовых строк в целые числа)

Эти команды преобразуют битовые строки в целочисленные значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_TO_*	Группа преобразования битовых строк в целые числа	FUN	 "**" — тип данных, относящийся к битовым строкам. "***" — целочисленный тип данных.	Out:=**_TO_*** (In); "**" — тип данных, относящийся к битовым строкам. "***" — целочисленный тип данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

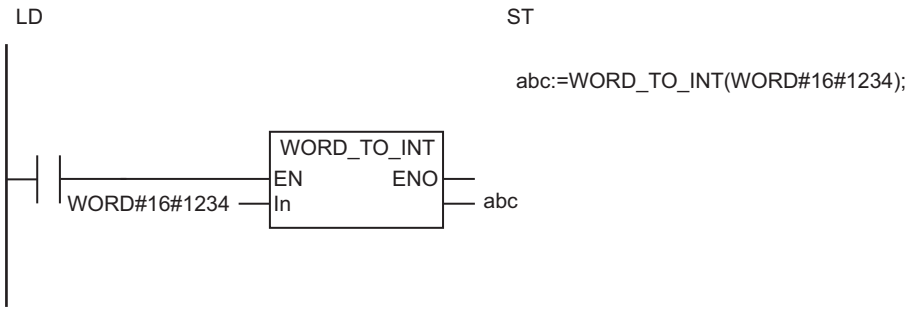
*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-309 ниже.

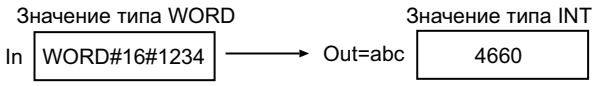
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		ОК	ОК	ОК	ОК															
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК							

Функция

Эти команды преобразуют битовую строку *In* в целочисленное значение.
Имя команды определяется типами данных *In* и *Out* (результат преобразования). Например, если *In* относится к типу данных WORD, а *Out* относится к типу данных INT, команда будет иметь имя WORD_TO_INT.

Ниже показан пример для команды WORD_TO_INT, в котором *In* = WORD #16#1234.





Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
BYTE	USINT	16#00...16#FF	0...255
	UINT		
	UDINT		
	ULINT		
	SINT		-128...127
	INT		
	DINT		
	LINT		
WORD	USINT	16#00...16#FF	0...255
	UINT	16#0000...16#FFFF	0...65535
	UDINT		
	ULINT		
	SINT	16#00...16#FF	-128...127
	INT	16#0000...16#FFFF	-32768...32767
	DINT		
	LINT		
DWORD	USINT	16#00...16#FF	0...255
	UINT	16#0000...16#FFFF	0...65535
	UDINT	16#0000_0000...16#FFFF_FFFF	0...4294967295
	ULINT		
	SINT	16#00...16#FF	-128...127
	INT	16#0000...16#FFFF	-32768...32767
	DINT	16#0000_0000...16#FFFF_FFFF	-2147483648...2147483647
	LINT		

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
LWORD	USINT	16#00...16#FF	0...255
	UINT	16#0000...16#FFFF	0...65535
	UDINT	16#0000_0000...16#FFFF_FFFF	0...4294967295
	ULINT	16#0000_0000_0000_0000...16#FFFF_FFFF_FFFF_FFFF	0...18446744073709551645
	SINT	16#00...16#FF	-128...127
	INT	16#0000...16#FFFF	-32768...32767
	DINT	16#0000_0000...16#FFFF_FFFF	-2147483648...2147483647
	LINT	16#0000_0000_0000_0000...16#FFFF_FFFF_FFFF_FFFF	-9223372036854775808...9223372036854775807

Дополнительная информация

- Для преобразования целого числа в битовую строку используйте команду ***_TO_*** (Группа преобразования целых чисел в битовые строки)* на стр. 2-302.
- Для преобразования значения любого типа данных в битовую строку используйте команду *TO_** (Группа преобразования в битовые строки)* на стр. 2-357.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.
- Если размер (разрядность) значений *Out* превышает размер (разрядность) значений *In*, старшие разряды *Out* будут содержать 0.
- Если разрядность *Out* меньше разрядности *In*, то старшие разряды будут отброшены.

_TO_* (Группа преобразования битовых строк в битовые строки)

Эти команды служат для преобразования битовых строк одного типа в битовые строки другого типа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_TO_*	Группа преобразования битовых строк в битовые строки	FUN	(@)**_TO_*** EN ENO In Out "**" и "***" — отличающиеся типы данных, относящиеся к битовым строкам.	Out:=**_TO_*** (In); "**" и "***" — отличающиеся типы данных, относящиеся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-312 ниже.

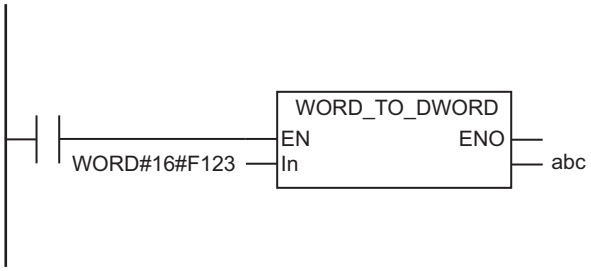
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Out		OK	OK	OK	OK															

Функция

Эти команды служат для преобразования битовой строки *In* одного типа в битовую строку другого типа.
Имя команды определяется типами данных *In* и *Out* (результат преобразования). Например, если *In* относится к типу данных WORD, а *Out* относится к типу данных DWORD, команда будет иметь имя WORD_TO_DWORD.

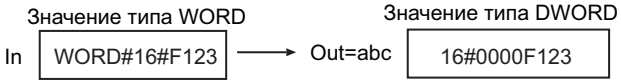
Ниже показан пример для команды WORD_TO_DWORD, в котором *In* = WORD#16#F123.

LD



ST

```
abc:=WORD_TO_DWORD(WORD#16#F123);
```



Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In* и *Out*
BYTE	WORD	16#00...16#FF
	DWORD	
	LWORD	
WORD	BYTE	16#00...16#FF
	DWORD	16#0000...16#FFFF
	LWORD	
DWORD	BYTE	16#00...16#FF
	WORD	16#0000...16#FFFF
	LWORD	16#0000_0000...16#FFFF_FFFF
LWORD	BYTE	16#00...16#FF
	WORD	16#0000...16#FFFF
	DWORD	16#0000_0000...16#FFFF_FFFF

Дополнительная информация

Для преобразования значения любого типа данных в битовую строку используйте команду *TO_** (Группа преобразования в битовые строки)* на стр. 2-357.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.
- Если размер (разрядность) значений *Out* превышает размер (разрядность) значений *In*, старшие разряды *Out* будут содержать 0.
- Если разрядность *Out* меньше разрядности *In*, то старшие разряды будут отброшены.

_TO_* (Группа преобразования битовых строк в вещественные числа)

Эти команды преобразуют битовые строки в вещественные значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_TO_*	Группа преобразования битовых строк в вещественные числа	FUN	(@)**_TO_*** EN ENO In Out "**" — тип данных, относящийся к битовым строкам. "***" — вещественный тип данных.	Out:=**_TO_*** (In); "**" — тип данных, относящийся к битовым строкам. "***" — вещественный тип данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-314 ниже.

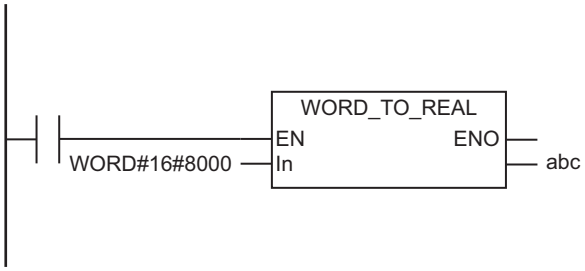
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Out														OK	OK					

Функция

Эти команды принимают битовую строку *In* в качестве целого числа без знака того же размера и преобразуют ее в вещественное число.
Имя команды определяется типами данных *In* и *Out* (результат преобразования). Например, если *In* относится к типу данных WORD, а *Out* относится к типу данных REAL, команда будет иметь имя WORD_TO_REAL.

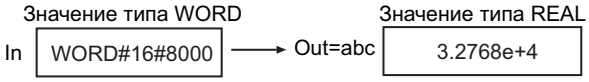
Ниже показан пример для команды WORD_TO_REAL, в котором *In* = WORD#16#8000.

LD



ST

```
abc:=WORD_TO_REAL(WORD#16#8000);
```



Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
BYTE	REAL	16#00...16#FF	0...2,55e+2
	LREAL		
WORD	REAL	16#0000...16#FFFF	0...6,5535e+4
	LREAL		
DWORD	REAL	16#0000_0000...16#FFFF_FFFF	0...4,294967e+9
	LREAL		0...4,294967295e+9
LWORD	REAL	16#0000_0000_0000_0000...16#FFFF_FFFF_FFFF_FFFF	0...1,844674e+19
	LREAL		0...1,84467440737095e+19

Дополнительная информация

- Для преобразования вещественного числа в битовую строку используйте команду ***_TO_*** (Группа преобразования вещественных чисел в битовые строки)* на стр. 2-318.
- Для преобразования значения любого типа данных в вещественное число используйте команду *TO_** (Группа преобразования в вещественные числа)* на стр. 2-359.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.
- В значащих разрядах выходного вещественного числа производится округление в зависимости от типов данных переменных *In* и *Out*. Из-за этого выходное значение может не быть точно равно входному значению. В следующей таблице перечислены комбинации типов данных, приводящие к возникновению ошибки.

Тип данных *In*	Тип данных *Out*	Значения, при которых возникает ошибка
DWORD	REAL	16#0100_0000 или больше
LWORD	LREAL	16#0002_0000_0000_0000 или больше

_TO_* (Группа преобразования вещественных чисел в целые числа)

Эти команды преобразуют вещественные значения в целочисленные значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_TO_*	Группа преобразования вещественных чисел в целые числа	FUN	(@)**_TO_*** EN ENO In Out "**" — вещественный тип данных "***" — целочисленный тип данных.	Out:=**_TO_*** (In); "**" — вещественный тип данных "***" — целочисленный тип данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-316 ниже.

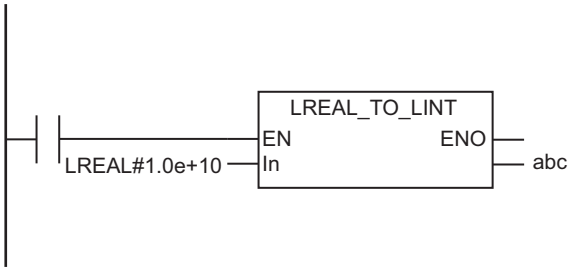
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														OK	OK					
Out						OK	OK	OK	OK	OK	OK	OK	OK							

Функция

Эти команды преобразуют вещественное значение *In* в целочисленное значение.
Имя команды определяется типами данных *In* и *Out* (результат преобразования). Например, если *In* относится к типу данных LREAL, а *Out* относится к типу данных LINT, команда будет иметь имя LREAL_TO_LINT.

Ниже показан пример для команды LREAL_TO_LINT, в котором *In* = LREAL#1.0e+10.

LD



ST

```
abc:=LREAL_TO_LINT(LREAL#1.0e+10);
```

Дробная часть значения *In*

Дробная часть значения *In* отбрасывается, значение округляется до ближайшего целого числа. В следующей таблице показано, как округляются значения.

Значение дробной части	Описание	Примеры
Меньше 0,5	Дробная часть отбрасывается.	1,49 → 1 -1,49 → -1
0,5	Если разряд единиц содержит четное число, дробная часть отбрасывается. Если это нечетное число, то значение округляется в большую сторону.	1,50 → 2 2,50 → 2 -1,50 → -2 -2,50 → -2
Больше 0,5	Дробная часть отбрасывается, число округляется в большую сторону.	1,51 → 2 -1,51 → -2

Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
REAL	USINT	0...2,55e+2	0...255
	UINT	0...6,5535e+4	0...65535
	UDINT	0...4,294967e+9	0...4294967295
	ULINT	0...1,844674e+19	0...18446744073709551615
	SINT	-1,28e+2...1,27e+2	-128...127
	INT	-3,2768e+4...3,2767e+4	-32768...32767
	DINT	-2,147483e+9...2,147483e+9	-2147483648...2147483647
	LINT	-9,223372e+18...9,223372e+18	-9223372036854775808...9223372036854775807

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In*	Допустимый диапазон для *Out*
LREAL	USINT	0...2,55e+2	0...255
	UINT	0...6,5535e+4	0...65535
	UDINT	0...4,294967295e+9	0...4294967295
	ULINT	0...1,84467440737095e+19	0...18446744073709551615
	SINT	-1,28e+2...1,27e+2	-128...127
	INT	-3,2768e+4...3,2767e+4	-32768...32767
	DINT	-2,147483648e+9...2,147483647e+9	-2147483648...2147483647
	LINT	-9,22337203685477e +18...9,22337203685477e+18	-9223372036854775808...9223372036854775807

Дополнительная информация

- Для преобразования целого числа в вещественное число используйте команду ***_TO_*** *(Группа преобразования целых чисел в вещественные числа)* на стр. 2-305.
- Для преобразования значения любого типа данных в целочисленное значение используйте команду *TO_*** *(Группа преобразования в целые числа)* на стр. 2-354.
- Для преобразования вещественного числа в целое число можно использовать одну из следующих команд: TRUNC (Усечение), Round (Округление вещественного числа) и RoundUp (Округление вещественного числа к большему).
  Все эти команды имеют вход типа REAL и выход типа DINT либо вход типа LREAL и выход типа LINT.
  Различия между этими командами отражены в следующей таблице.

Входное значение	Выходное значение			
	REAL_TO_INT	TRUNC	Round	RoundUp
REAL#1.6	INT#2	DINT#1	DINT#2	DINT#2
REAL#1.5	INT#2	DINT#1	DINT#2	DINT#2
REAL#1.5	INT#1	DINT#1	DINT#1	DINT#2
REAL#2.5	INT#2	DINT#2	DINT#2	DINT#3
REAL#-1.6	INT#-2	DINT#-1	DINT#-2	DINT#-2
REAL#-1.5	INT#-2	DINT#-1	DINT#-2	DINT#-2
REAL#-1.4	INT#-1	DINT#-1	DINT#-1	DINT#-2
REAL#-2.5	INT#-2	DINT#-2	DINT#-2	DINT#-3

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.
- Если результат преобразования выйдет за диапазон допустимых значений *Out*, *Out* будет содержать неопределенное значение. Всегда следите за тем, чтобы значение *In* находилось в пределах допустимого диапазона, чтобы результат преобразования не вышел за диапазон допустимых значений *Out*.

_TO_* (Группа преобразования вещественных чисел в битовые строки)

Эти команды преобразуют вещественные значения в битовые строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_TO_*	Группа преобразования вещественных чисел в битовые строки	FUN	(@)**_TO_*** EN ENO In Out "**" — вещественный тип данных "***" — тип данных, относящийся к битовым строкам.	Out:=**_TO_*** (In); "**" — вещественный тип данных "***" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	0
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

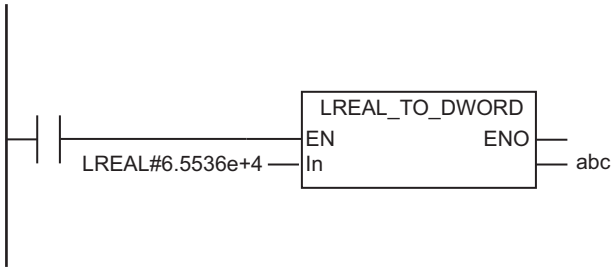
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														ОК	ОК					
Out		ОК	ОК	ОК	ОК															

Функция

Эти команды преобразуют вещественное значение *In* в битовую строку.
Имя команды определяется типами данных *In* и *Out* (выход преобразования). Например, если *In* относится к типу данных LREAL, а *Out* относится к типу данных DWORD, команда будет иметь имя LREAL_TO_DWORD.

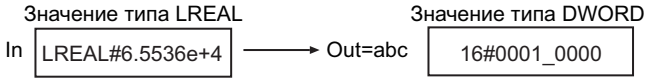
Ниже показан пример для команды LREAL_TO_DWORD, в котором *In* = LREAL#6.5536e+4.

LD



ST

```
abc:=LREAL_TO_DWORD(LREAL#6.5536e+4);
```



В следующей таблице приведены некоторые примеры преобразования.

Значение *In*	Целое	Значение *Out*
1,6	2	16#0002
3,5	4	16#0004

Процедура преобразования

Преобразование выполняется описанным ниже образом.

1 Значение *In* округляется до ближайшего целого числа описанным ниже образом.

2 Полученное целое число принимается за целое число без знака и выводится в виде битовой строки.

Округление

В следующей таблице показано, как округляются значения.

Значение дробной части	Описание	Примеры
Меньше 0,5	Дробная часть отбрасывается.	1,49 → 1
0,5	Если разряд единиц содержит четное число, дробная часть отбрасывается. Если это нечетное число, то значение округляется в большую сторону.	1,50 → 2 2,50 → 2
Больше 0,5	Дробная часть отбрасывается, число округляется в большую сторону.	1,51 → 2

Дополнительная информация

Для преобразования битовой строки в вещественное число используйте команду ****_TO_**** *(Группа преобразования битовых строк в вещественные числа)* на стр. 2-313.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.

- Если результат преобразования выйдет за диапазон допустимых значений *Out*, *Out* будет содержать неопределенное значение. Всегда следите за тем, чтобы значение *In* находилось в пределах допустимого диапазона, чтобы результат преобразования не вышел за диапазон допустимых значений *Out*.
- В случае ввода отрицательного значения результат преобразования зависит от модели модуля ЦПУ. Если вводятся отрицательные значения, перед вводом системы в эксплуатацию необходимо тщательно протестировать работу программы.

_TO_* (Группа преобразования вещественных чисел в вещественные числа)

Эти команды служат для преобразования вещественных чисел одного типа в вещественные числа другого типа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
_TO_*	Группа преобразования вещественных чисел в вещественные числа	FUN	(@)**_TO_*** EN ENO In Out "**" и "***" — отличающиеся вещественные типы данных.	Out:=**_TO_*** (In); "**" и "***" — отличающиеся вещественные типы данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-322 ниже.

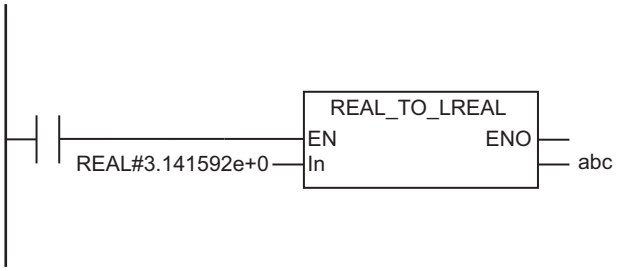
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														OK	OK					
Out														OK	OK					

Функция

Эти команды служат для преобразования вещественного числа *In* одного типа в вещественное число другого типа.
Имя команды определяется типами данных *In* и *Out* (результат преобразования). Например, если *In* относится к типу данных REAL, а *Out* относится к типу данных LREAL, команда будет иметь имя REAL_TO_LREAL.

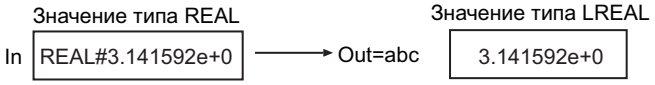
Ниже показан пример для команды REAL_TO_LREAL, в котором *In* = REAL#3.141592e+0.

LD



ST

```
abc:=REAL_TO_LREAL(REAL#3.141592e+0);
```



Допустимый диапазон

В следующей таблице приведены диапазоны допустимых значений для переменных *In* и *Out* в соответствии с их типами данных.

Тип данных *In*	Тип данных *Out*	Допустимый диапазон для *In* и *Out*
REAL	LREAL	-3,402823e+38...3,402823e+38 или +∞/-∞
LREAL	REAL	

Дополнительная информация

Для преобразования значения любого типа данных в вещественное число используйте команду *TO_** (Группа преобразования в вещественные числа)* на стр. 2-359.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типам данных переменных *In* и *Out*.
- Если значение *In* является положительной или отрицательной бесконечностью, значение *Out* также является положительной или отрицательной бесконечностью.
- Если значение *In* является нечисловым, то и значение *Out* также является нечисловым.
- Если результат преобразования выходит за диапазон допустимых значений *Out*, значение *Out* является бесконечностью с тем же знаком, что и у значения *In*.
- Для команды LREAL_TO_REAL: если значение *In* ближе к 0, чем ±1,175494e-38, значение *Out* будет равно 0.

**_TO_STRING (Группа преобразования целых чисел в текстовые строки)

Эти команды преобразуют целочисленные значения в текстовые строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
**_TO_STRING	Группа преобразования целых чисел в текстовые строки	FUN	(@)**_TO_STRING EN ENO In Out "**" — целочисленный тип данных.	Out:=**_TO_STRING(In); "**" — целочисленный тип данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазон допустимых значений зависит от типа данных *In*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-324.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						OK	OK	OK	OK	OK	OK	OK	OK							
Out																				OK

Функция

Эти команды преобразуют целочисленное значение *In* в тестовую строку.
Число, указанное в *In*, выводится в результат преобразования *Out* в виде текстовой строки. В конец значения *Out* добавляется нулевой символ (NULL) (16#00).

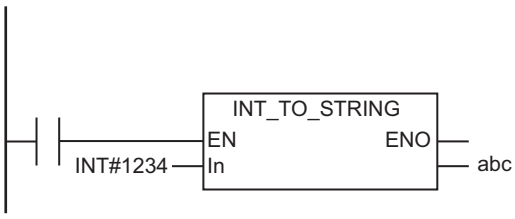
Текст в *Out* выравнивается по левому краю.

Если число значащих разрядов в *In* меньше числа разрядов, предполагаемого типом данных *In*, ведущие нули этого значения в *Out* не выводятся. Другими словами, ведущие нули отбрасываются.

Если *In* содержит отрицательное значение, то в начале текстовой строки добавляется знак минус (-).
Имя команды определяется типом данных *In*. Например, если *In* относится к типу данных INT, команда будет иметь имя INT_TO_STRING.

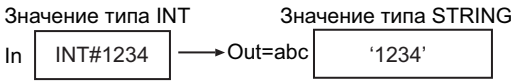
Ниже показан пример для команды INT_TO_STRING, в котором *In* = INT#1234.

LD



ST

```
abc:=INT_TO_STRING(INT#1234);
```



Допустимый диапазон

Диапазон допустимых значений *Out* зависит от типа данных *In*, что показано ниже:

Тип данных *In*	Диапазон допустимых значений *Out* (максимальное количество байтов)
USINT	4 байта (3 однобайтовых буквенно-цифровых символа + последний символ NULL)
UINT	6 байтов (5 однобайтовых буквенно-цифровых символов + последний символ NULL)
UDINT	11 байтов (10 однобайтовых буквенно-цифровых символов + последний символ NULL)
ULINT	21 байт (20 однобайтовых буквенно-цифровых символов + последний символ NULL)
SINT	5 байтов (4 однобайтовых буквенно-цифровых символа + последний символ NULL)
INT	7 байтов (6 однобайтовых буквенно-цифровых символов + последний символ NULL)
DINT	12 байтов (11 однобайтовых буквенно-цифровых символов + последний символ NULL)
LINT	21 байт (20 однобайтовых буквенно-цифровых символов + последний символ NULL)

Дополнительная информация

Для преобразования текстовой строки в целое число используйте команду *STRING_TO_** *(Группа преобразования текстовых строк в целые числа)* на стр. 2-344.

Меры предосторожности для обеспечения надлежащей эксплуатации

Всегда используйте правильное имя команды, соответствующее типу данных переменной *In*.

**_TO_STRING (Группа преобразования битовых строк в текстовые строки)

Эти команды преобразуют битовые строки в текстовые строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
**_TO_STRING	Группа преобразования битовых строк в текстовые строки	FUN	(@)**_TO_STRING EN ENO In Out "**" — тип данных, относящийся к битовым строкам.	Out:=**_TO_STRING(In); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазон допустимых значений зависит от типа данных *In*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-326.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Out																				OK

Функция

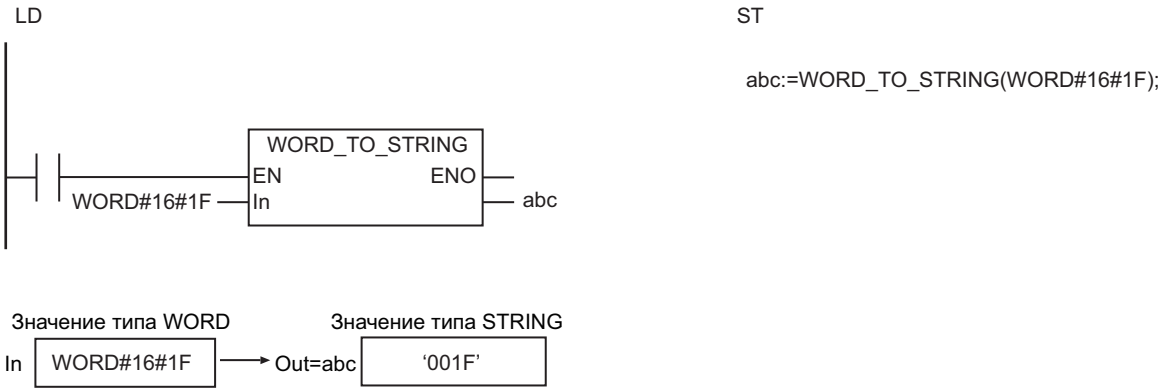
Эти команды преобразуют битовую строку *In* в текстовую строку.
Шестнадцатеричное число, указанное в *In*, выводится в результат преобразования *Out* в виде текстовой строки. Префикс шестнадцатеричного числа "#16" в *Out* не выводится. В конец значения *Out* добавляется нулевой символ (NULL) (16#00).

Текст в *Out* выравнивается по левому краю.

Если для значения *In* требуется меньше разрядов, чем предусмотрено типом данных *In*, старшие разряды *Out* будут содержать "0". Другими словами, неиспользуемые разряды заполняются

нулями. Количество байтов в значении *Out* (включая символ NULL) всегда будет на единицу больше, чем удвоенное количество байтов в значении *In*.
Имя команды определяется типом данных *In*. Например, если *In* относится к типу данных WORD, команда будет иметь имя WORD_TO_STRING.

Ниже показан пример для команды WORD_TO_STRING, в котором *In* = WORD#16#1F.



Допустимый диапазон

Диапазон допустимых значений *Out* зависит от типа данных *In*, что показано ниже:

Тип данных *In*	Диапазон допустимых значений *Out* (максимальное количество байтов)
BYTE	3 байта (2 однобайтовых буквенно-цифровых символа + последний символ NULL)
WORD	5 байтов (4 однобайтовых буквенно-цифровых символа + последний символ NULL)
DWORD	9 байтов (8 однобайтовых буквенно-цифровых символов + последний символ NULL)
LWORD	17 байтов (16 однобайтовых буквенно-цифровых символов + последний символ NULL)

Дополнительная информация

Чтобы преобразовать значение *In* в текстовую строку со знаком, сначала преобразуйте его в целое число со знаком, используя команду ***_TO_*** (Группа преобразования битовых строк в целые числа)* на стр. 2-308, а затем воспользуйтесь командой ***_TO_STRING (Группа преобразования целых чисел в текстовые строки)* на стр. 2-323.

Меры предосторожности для обеспечения надлежащей эксплуатации

Всегда используйте правильное имя команды, соответствующее типу данных переменной *In*.

**_TO_STRING (Группа преобразования вещественных чисел в текстовые строки)

Эти команды преобразуют вещественные значения в текстовые строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
**_TO_STRING	Группа преобразования вещественных чисел в текстовые строки	FUN	(@)**_TO_STRING EN ENO In Out "**" — вещественный тип данных	Out:=**_TO_STRING(In); "**" — вещественный тип данных

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	0,0
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

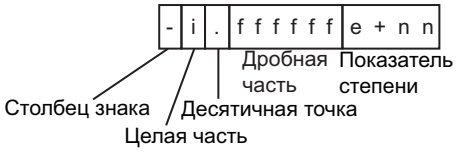
*1. Диапазон допустимых значений зависит от типа данных *In*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-328.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														OK	OK					
Out																				OK

Функция

Эти команды преобразуют вещественное значение *In* в текстовую строку.
Значение *In* выражается в виде буквенно-цифровой текстовой строки и выводится в результат преобразования *Out*.

Для *Out* используется следующий формат представления:

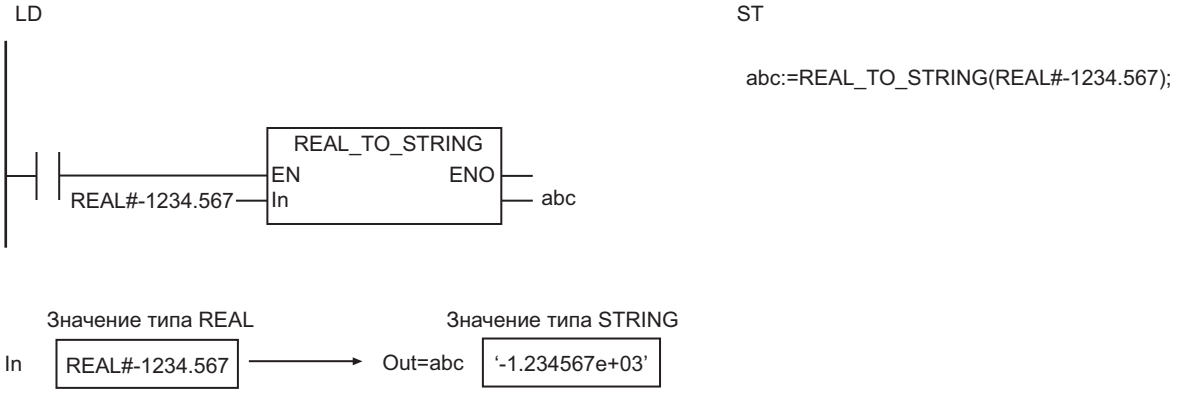


Параметр	Описание
Столбец знака	Если *In* содержит отрицательное значение, то добавляется знак минус (-). Если *In* содержит положительное значение, знак плюс (+) не добавляется.
Целая часть	Целая часть всегда состоит только из одной цифры.
Десятичная точка	Десятичная точка всегда имеется, даже если *In* не является десятичным числом.
Дробная часть	Если значение *In* относится к типу данных REAL, то выводится 6 разрядов, а если к типу данных LREAL, то 14 разрядов.
Показатель степени	Показатель степени всегда имеется. "nn" — это 2 или 3 цифры. Знак "nn" положителен (+), если абсолютное значение *In* равно 1,0 или больше, и отрицателен (-), если оно меньше 1,0.

В конец значения *Out* добавляется нулевой символ (NULL) (16#00).

Имя команды определяется типом данных *In*. Например, если *In* относится к типу данных REAL, команда будет иметь имя REAL_TO_STRING.

Ниже показан пример для команды REAL_TO_STRING, в котором *In* = REAL#-1234.567.



Допустимый диапазон

В таблице ниже приведены значения *Out* для случаев, когда *In* равно 0 либо является бесконечностью или нечисловым значением.

Значение *In*	Значение *Out*
0	«0»
+∞	«inf»
-∞	«-inf»
Нечисловое значение	«nan» или «-nan»

Дополнительная информация

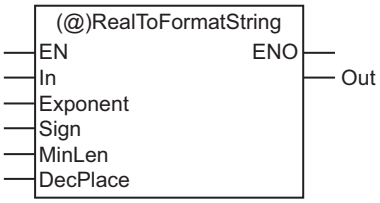
- Для преобразования текстовой строки в вещественное число используйте команду *STRING_TO_** (Группа преобразования текстовых строк в вещественные числа)* на стр. 2-350.
- Если нужно указать формат при преобразовании вещественного числа в текстовую строку, используйте команду *RealToFormatString* на стр. 2-330 или *LrealToFormatString* на стр. 2-337.

Меры предосторожности для обеспечения надлежащей эксплуатации

Всегда используйте правильное имя команды, соответствующее типу данных переменной *In*.

RealToFormatString

Команда RealToFormatString преобразует переменную типа REAL в текстовую строку с указанным форматом.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
RealToFormat String	REAL в форматированную текстовую строку	FUN		Out:=RealToFormatString(In, Exponent, Sign, MinLen, DecPlace);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	0,0
Exponent	Показатель степени		ИСТИНА: показатель степени ЛОЖЬ: без показателя степени			ЛОЖЬ
Sign	Столбец знака		ИСТИНА: столбец знака ЛОЖЬ: без столбца знака			
MinLen	Минимальное число разрядов		Минимальное число разрядов в *Out*			6
DecPlace	Точность		Число разрядов дробной части в *Out*	0...15		
Out	Результат преобразования	Выход	Результат преобразования	Макс. 327 байт (326 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														ОК						
Exponent	ОК																			
Sign	ОК																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MinLen						OK														
DecPlace						OK														
Out																				OK

Функция

Команда RealToFormatString преобразует переменную *In* типа REAL в текстовую строку. Значение *In* выражается в виде буквенно-цифровой текстовой строки и выводится в результат преобразования *Out*. В конец значения *Out* добавляется нулевой символ (NULL) (16#00).

Если *In* содержит отрицательное значение, то в начале текстовой строки добавляется знак минус (-). Если *In* содержит положительное значение, знак плюс (+) в начало текстовой строки не добавляется.

Формат *Out* определяется показателем степени *Exponent*, столбцом знака *Sign*, минимальным числом разрядов *MinLen* и точностью *DecPlace*.

Входная переменная	Описание
Exponent	Параметр *Exponent* указывает, должно ли выходное значение содержать показатель степени. ИСТИНА: показатель степени ЛОЖЬ: без показателя степени
Sign	Параметр *Sign* указывает, должно ли выходное значение содержать столбец знака. ИСТИНА: столбец знака ЛОЖЬ: без столбца знака Столбец знака используется только для знака минус (-). Если столбец знака указан, а число является положительным, то столбец знака будет содержать символ пробела. Если столбец знака не указан, а число является отрицательным, знак минус (-) будет добавлен в начало целой части. В то же время, если число разрядов в результате преобразования превосходит значение *MinLen* и результат преобразования является положительным, старший разряд помещается в столбец знака.

Входная переменная	Описание
MinLen	*MinLen* — это минимальное общее число разрядов, используемых для столбца знака, целой части, десятичной точки, дробной части и показателя степени. Если результат преобразования содержит меньше разрядов, чем значение *MinLen*, текстовая строка будет выровнена по правому краю (за исключением столбца знака), а оставшиеся разряды будут содержать символы пробела. Если число разрядов в результате преобразования превышает значение *MinLen*, то текстовая строка выравнивается по левому краю, и переменной *Out* присваивается текстовая строка, соответствующая разрядам, которые выходят за значение *MinLen*.
DecPlace	*DecPlace* — это количество разрядов в дробной части. Если количество разрядов превышает значение *DecPlace*, лишние разряды дробной части отбрасываются и число округляется, как описано ниже. Если *DecPlace* = *0*, дробная часть и десятичная точка не указываются.

В таблице ниже приведены значения *Out* для случаев, когда *In* является бесконечностью или нечисловым значением.

Значение *In*	Значение *Out*
+∞	«inf»
-∞	«-inf»
Нечисловое значение	«nan» или «-nan»

Округление

В следующей таблице показано, как округляются значения.

Значение дробной части	Описание	Примеры
Меньше 0,5	Дробная часть отбрасывается.	1,49 → 1
0,5	Если разряд единиц содержит четное число, дробная часть отбрасывается. Если это нечетное число, то значение округляется в большую сторону.	1,50 → 2 2,50 → 2
Больше 0,5	Дробная часть отбрасывается, число округляется в большую сторону.	1,51 → 2

Входные и выходные переменные

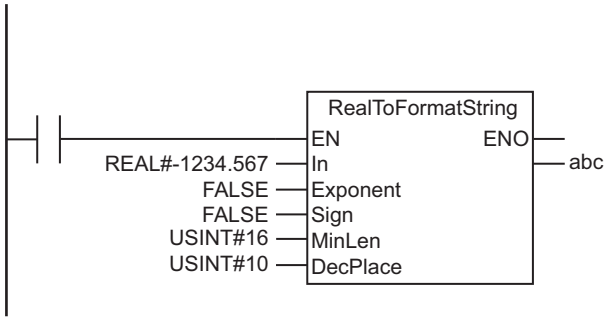
В следующих примерах показано, как входные значения преобразуются в значение *Out*.

● Пример 1

Переменные	Настройка
In	REAL#-1234.567
Exponent	ЛОЖЬ
Sign	ЛОЖЬ
MinLen	USINT#16
DecPlace	USINT#10

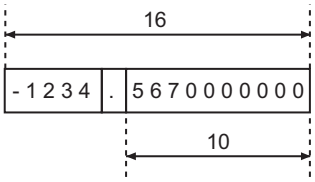
В данном примере для отрицательного числа не указан столбец знака, поэтому знак минус (-) добавляется в начало целой части.

LD



ST

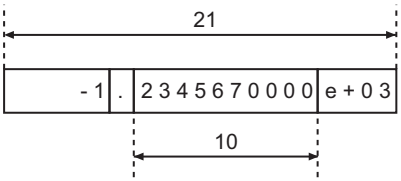
```
abc:=RealToFormatString(REAL#-1234.567, FALSE,
                        FALSE, USINT#16,
                        USINT#10);
```



● Пример 2

Переменные	Настройка
In	REAL#-1234.567
Exponent	ИСТИНА
Sign	ЛОЖЬ
MinLen	USINT#21
DecPlace	USINT#10

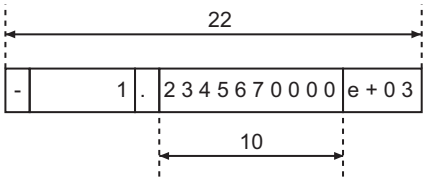
В данном примере значение *MinLen* превышает число разрядов в текстовой строке, поэтому текстовая строка выравнивается по правому краю и перед ней добавляются символы пробела.



● Пример 3

Переменные	Настройка
In	REAL#-1234.567
Exponent	ИСТИНА
Sign	ИСТИНА
MinLen	USINT#22
DecPlace	USINT#10

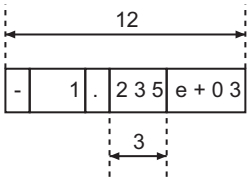
Столбец знака всегда находится слева. Перед целой частью добавляются символы пробела.



● Пример 4

Переменные	Настройка
In	REAL#-1234.567
Exponent	ИСТИНА
Sign	ИСТИНА
MinLen	USINT#12
DecPlace	USINT#3

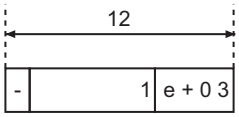
Число округляется до третьего разряда дробной части, так как *DecPlace* = USINT#3.



● Пример 5

Переменные	Настройка
In	REAL#-1234.567
Exponent	ИСТИНА
Sign	ИСТИНА
MinLen	USINT#12
DecPlace	USINT#0

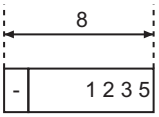
Число округляется до единиц, так как *DecPlace* = USINT#0. Десятичная точка также не указывается.



● Пример 6

Переменные	Настройка
In	REAL#-1234.567
Exponent	ЛОЖЬ
Sign	ИСТИНА
MinLen	USINT#8
DecPlace	USINT#0

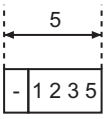
В данном примере показатель степени не указан, а целая часть состоит только из четырех цифр. Число округляется до единиц.



● Пример 7

Переменные	Настройка
In	REAL#-1234.567
Exponent	ЛОЖЬ
Sign	ИСТИНА
MinLen	USINT#2
DecPlace	USINT#0

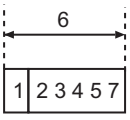
В данном примере число разрядов в целой части *In* (четыре разряда) больше, чем значение *MinLen* (USINT#2). Указываются четыре разряда целой части.



● Пример 8

Переменные	Настройка
In	REAL#123456.7
Exponent	ЛОЖЬ
Sign	ИСТИНА
MinLen	USINT#4
DecPlace	USINT#0

В данном примере число разрядов в целой части *In* (шесть разрядов) больше, чем значение *MinLen* (USINT#4). Указываются шесть разрядов целой части. Значение *In* положительно, поэтому самый старший разряд помещается в столбец знака.



Дополнительная информация

- Параметры *Exponent*, *Sign*, *MinLen* и *DecPlace* могут быть опущены. Вместо любой опущенной входной переменной применяется значение по умолчанию.
- Для преобразования переменной типа LREAL в текстовую строку используйте команду *LrealToFormatString* на стр. 2-337.
- Для преобразования текстовой строки в вещественное число используйте команду *STRING_TO_** (Группа преобразования текстовых строк в вещественные числа)* на стр. 2-350.

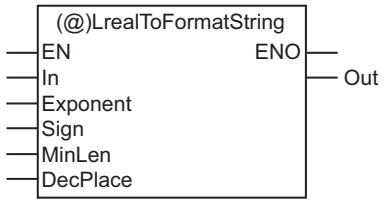
Меры предосторожности для обеспечения надлежащей эксплуатации

В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Значение *DecPlace* находится за пределами допустимого диапазона.
- Значение *DecPlace* больше значения *MinLen*.

LrealToFormatString

Команда LrealToFormatString преобразует переменную типа LREAL в текстовую строку с указанным форматом.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LrealToFormat String	LREAL в форматированную текстовую строку	FUN		Out:=LrealToFormatString (In, Exponent, Sign, MinLen, DecPlace);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	0,0
Exponent	Показатель степени		ИСТИНА: показатель степени ЛОЖЬ: без показателя степени			ЛОЖЬ
Sign	Столбец знака		ИСТИНА: столбец знака ЛОЖЬ: без столбца знака			
MinLen	Минимальное число разрядов		Минимальное число разрядов в *Out*			6
DecPlace	Точность		Число разрядов дробной части в *Out*	0...15		
Out	Результат преобразования	Выход	Результат преобразования	Макс. 327 байт (326 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In															OK					
Exponent	OK																			
Sign	OK																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MinLen						OK														
DecPlace						OK														
Out																				OK

Функция

Команда LrealToFormatString преобразует переменную *In* типа LREAL в текстовую строку. Значение *In* выражается в виде буквенно-цифровой текстовой строки и выводится в результат преобразования *Out*. В конец значения *Out* добавляется нулевой символ (NULL) (16#00).

Если *In* содержит отрицательное значение, то в начале текстовой строки добавляется знак минус (-). Если *In* содержит положительное значение, знак плюс (+) в начало текстовой строки не добавляется.

Формат *Out* определяется показателем степени *Exponent*, столбцом знака *Sign*, минимальным числом разрядов *MinLen* и точностью *DecPlace*.

Входная переменная	Описание
Exponent	Параметр *Exponent* указывает, должно ли выходное значение содержать показатель степени. ИСТИНА: показатель степени ЛОЖЬ: без показателя степени
Sign	Параметр *Sign* указывает, должно ли выходное значение содержать столбец знака. ИСТИНА: столбец знака ЛОЖЬ: без столбца знака Столбец знака используется только для знака минус (-). Если столбец знака указан, а число является положительным, то столбец знака будет содержать символ пробела. Если столбец знака не указан, а число является отрицательным, знак минус (-) будет добавлен в начало целой части. В то же время, если число разрядов в результате преобразования превосходит значение *MinLen* и результат преобразования является положительным, старший разряд помещается в столбец знака.

Входная переменная	Описание
MinLen	*MinLen* — это минимальное общее число разрядов, используемых для столбца знака, целой части, десятичной точки, дробной части и показателя степени. Если результат преобразования содержит меньше разрядов, чем значение *MinLen*, текстовая строка будет выровнена по правому краю (за исключением столбца знака), а оставшиеся разряды будут содержать символы пробела. Если число разрядов в результате преобразования превышает значение *MinLen*, то текстовая строка выравнивается по левому краю, и переменной *Out* присваивается текстовая строка, соответствующая разрядам, которые выходят за значение *MinLen*.
DecPlace	*DecPlace* — это количество разрядов в дробной части. Если количество разрядов превышает значение *DecPlace*, лишние разряды дробной части отбрасываются и число округляется, как описано ниже. Если *DecPlace* = 0, дробная часть и десятичная точка не указываются.

В таблице ниже приведены значения *Out* для случаев, когда *In* является бесконечностью или нечисловым значением.

Значение *In*	Значение *Out*
+∞	«inf»
-∞	«-inf»
Нечисловое значение	«nan» или «-nan»

Округление

В следующей таблице показано, как округляются значения.

Значение дробной части	Описание	Примеры
Меньше 0,5	Дробная часть отбрасывается.	1,49 → 1
0,5	Если разряд единиц содержит четное число, дробная часть отбрасывается. Если это нечетное число, то значение округляется в большую сторону.	1,50 → 2 2,50 → 2
Больше 0,5	Дробная часть отбрасывается, число округляется в большую сторону.	1,51 → 2

Входные и выходные переменные

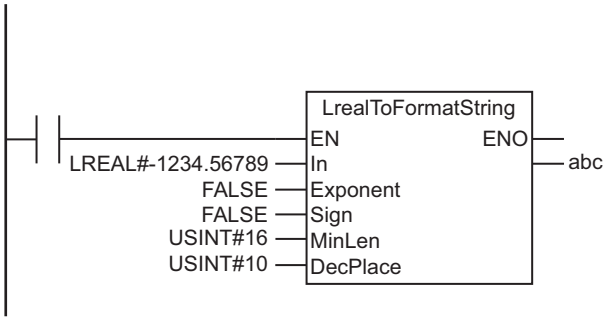
В следующих примерах показано, как входные значения преобразуются в значение *Out*.

● Пример 1

Переменные	Настройка
In	LREAL#-1234.56789
Exponent	ЛОЖЬ
Sign	ЛОЖЬ
MinLen	USINT#16
DecPlace	USINT#10

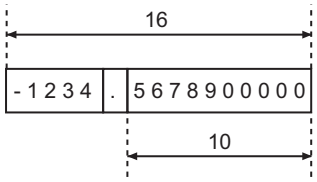
В данном примере для отрицательного числа не указан столбец знака, поэтому знак минус (-) добавляется в начало целой части.

LD



ST

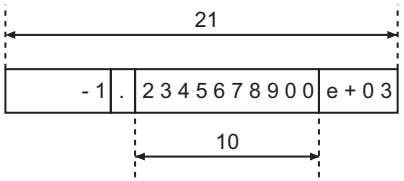
```
abc:=LrealToFormatString(LREAL#-1234.56789, FALSE,
                         FALSE, USINT#16,
                         USINT#10);
```



● Пример 2

Переменные	Настройка
In	LREAL#-1234.56789
Exponent	ИСТИНА
Sign	ЛОЖЬ
MinLen	USINT#21
DecPlace	USINT#10

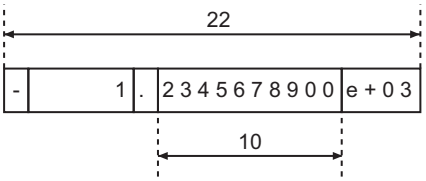
В данном примере значение *MinLen* превышает число разрядов в текстовой строке, поэтому текстовая строка выравнивается по правому краю и перед ней добавляются символы пробела.



● Пример 3

Переменные	Настройка
In	LREAL#-1234.56789
Exponent	ИСТИНА
Sign	ИСТИНА
MinLen	USINT#22
DecPlace	USINT#10

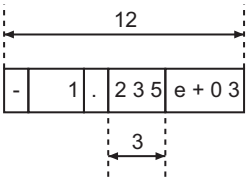
Столбец знака всегда находится слева. Перед целой частью добавляются символы пробела.



● Пример 4

Переменные	Настройка
In	LREAL#-1234.56789
Exponent	ИСТИНА
Sign	ИСТИНА
MinLen	USINT#12
DecPlace	USINT#3

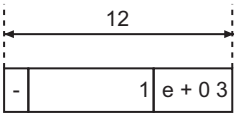
Число округляется до третьего разряда дробной части, так как *DecPlace* = USINT#3.



● Пример 5

Переменные	Настройка
In	LREAL#-1234.56789
Exponent	ИСТИНА
Sign	ИСТИНА
MinLen	USINT#12
DecPlace	USINT#0

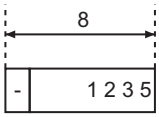
Число округляется до единиц, так как *DecPlace* = USINT#0. Десятичная точка также не указывается.



● Пример 6

Переменные	Настройка
In	LREAL#-1234.56789
Exponent	ЛОЖЬ
Sign	ИСТИНА
MinLen	USINT#8
DecPlace	USINT#0

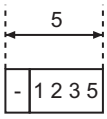
В данном примере показатель степени не указан, а целая часть состоит только из четырех цифр. Число округляется до единиц.



● Пример 7

Переменные	Настройка
In	LREAL#-1234.56789
Exponent	ЛОЖЬ
Sign	ИСТИНА
MinLen	USINT#2
DecPlace	USINT#0

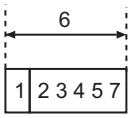
В данном примере число разрядов в целой части *In* (четыре разряда) больше, чем значение *MinLen* (USINT#2). Указываются четыре разряда целой части.



● Пример 8

Переменные	Настройка
In	LREAL#123456.789
Exponent	ЛОЖЬ
Sign	ИСТИНА
MinLen	USINT#4
DecPlace	USINT#0

В данном примере число разрядов в целой части *In* (шесть разрядов) больше, чем значение *MinLen* (USINT#4). Указываются шесть разрядов целой части. Значение *In* положительно, поэтому самый старший разряд помещается в столбец знака.



Дополнительная информация

- Параметры *Exponent*, *Sign*, *MinLen* и *DecPlace* могут быть опущены. Вместо любой опущенной входной переменной применяется значение по умолчанию.
- Для преобразования переменной типа REAL в текстовую строку используйте команду *RealToFormatString* на стр. 2-330.
- Для преобразования текстовой строки в вещественное число используйте команду *STRING_TO_** (Группа преобразования текстовых строк в вещественные числа)* на стр. 2-350.

Меры предосторожности для обеспечения надлежащей эксплуатации

В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Значение *DecPlace* находится за пределами допустимого диапазона.
- Значение *DecPlace* больше значения *MinLen*.

STRING_TO_** (Группа преобразования текстовых строк в целые числа)

Эти команды преобразуют текстовые строки в целочисленные значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
STRING_TO_**	Группа преобразования текстовых строк в целые числа	FUN	(@)STRING_TO_** EN ENO In Out "**" — целочисленный тип данных.	Out:=STRING_TO_** (In); "**" — целочисленный тип данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	"
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

*1. Диапазон допустимых значений зависит от типа данных *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-345.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Out						OK	OK	OK	OK	OK	OK	OK	OK							

Функция

Эти команды преобразуют текстовую строку *In* в целочисленное значение.

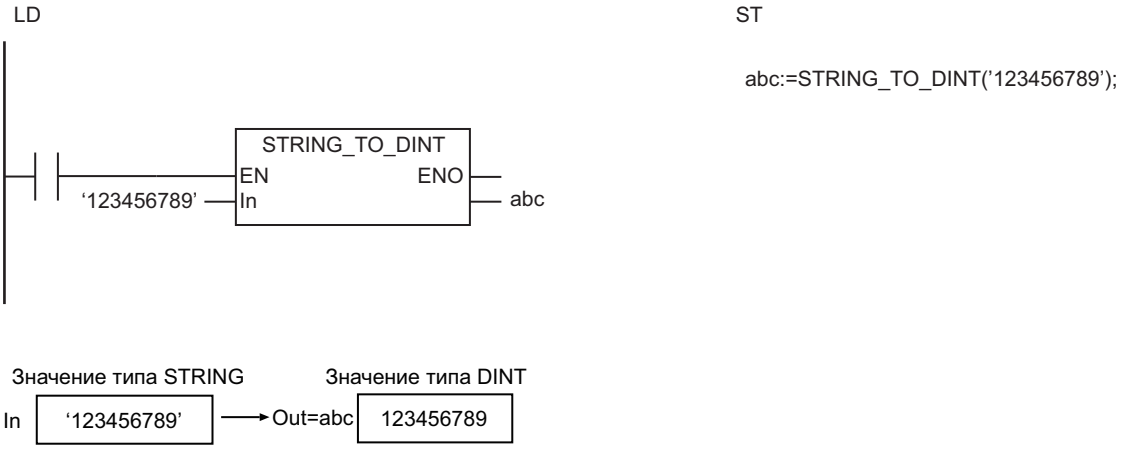
По существу, текстовая строка в *In* должна состоять только из цифр от «0» до «9». Допускаются указанные ниже исключения.

- Если первым символом в *In* является одиночный знак минуса (-) или одиночный знак плюса (+), он обрабатывается как знак.
- Символы пробела в начале *In* игнорируются.
- Символы пробела, расположенные между начальным знаком минуса (-) или плюса (+) и числом, игнорируются.

- Одиночные символы подчеркивания (_), расположенные в любом месте, игнорируются.
- Если в каком-либо месте строки имеются два или больше символов подчеркивания (_) подряд, возникает ошибка.
- Если строка начинается или заканчивается символом подчеркивания (_), возникает ошибка.
- Если между знаком минуса (-) или плюса (+) в начале строки и числом расположен символ подчеркивания (_), возникает ошибка

Имя команды определяется типом данных *Out* (результат преобразования). Например, если *Out* относится к типу данных DINT, команда будет иметь имя STRING_TO_DINT.

Ниже показан пример для команды STRING_TO_DINT, в котором *In* = «123456789».



Допустимый диапазон

Диапазон допустимых значений *In* зависит от типа данных *Out*, что показано ниже:

Тип данных *Out*	Диапазон допустимых значений *In* (максимальное количество байтов)*1
USINT	4 байта (3 однобайтовых буквенно-цифровых символа + последний символ NULL)
UINT	6 байтов (5 однобайтовых буквенно-цифровых символов + последний символ NULL)
UDINT	11 байтов (10 однобайтовых буквенно-цифровых символов + последний символ NULL)
ULINT	21 байт (20 однобайтовых буквенно-цифровых символов + последний символ NULL)
SINT	5 байтов (4 однобайтовых буквенно-цифровых символа + последний символ NULL)
INT	7 байтов (6 однобайтовых буквенно-цифровых символов + последний символ NULL)
DINT	12 байтов (11 однобайтовых буквенно-цифровых символов + последний символ NULL)
LINT	21 байт (20 однобайтовых буквенно-цифровых символов + последний символ NULL)

*1. Символы пробела ( ) и нули в начале текстовой строки, а также символы подчеркивания (_) в текстовой строке не включаются в число байтов.

Дополнительная информация

- Для преобразования текстовой строки в шестнадцатеричное число используйте команду *STRING_TO_** (Группа преобразования текстовых строк в битовые строки)* на стр. 2-347.
- Для преобразования целого числа в текстовую строку используйте команду *_TO_STRING (Группа преобразования целых чисел в текстовые строки)* на стр. 2-323.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типу данных переменной *Out*.
- Если значение *In* равно «-0», то значение *Out* равно *0*.
- В указанных ниже случаях происходит ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Текстовая строка в *In* не выражает число.
  b) Результат преобразования выходит за диапазон значений, допускаемых типом данных переменной *Out*.

STRING_TO_** (Группа преобразования текстовых строк в битовые строки)

Эти команды преобразуют текстовые строки в битовые строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
STRING_TO_**	Группа преобразования текстовых строк в битовые строки	FUN	(@)STRING_TO_** EN — ENO In — Out "**" — тип данных, относящийся к битовым строкам.	Out:=STRING_TO_** (In); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	"
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

*1. Диапазон допустимых значений зависит от типа данных *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-348.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				ОК
Out		ОК	ОК	ОК	ОК															

Функция

Эти команды интерпретируют содержимое текстовой строки *In* как шестнадцатеричное число и преобразуют его в битовую строку.

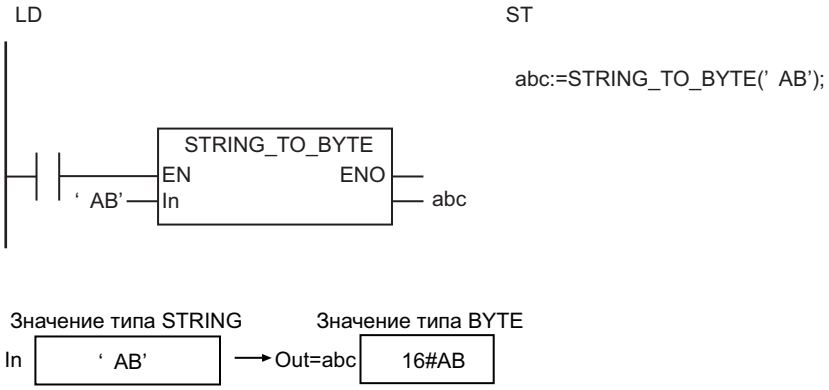
По существу, текстовая строка в *In* должна состоять только из символов от "0" до "9", от "a" до "f" и от "A" до "F". Допускаются указанные ниже исключения.

- Расположенные подряд символы пробела или нули в начале *In* игнорируются.
- Одиночные символы подчеркивания (_), расположенные в любом месте, игнорируются.

- Если в каком-либо месте строки имеются два или больше символов подчеркивания (_) подряд, возникает ошибка.
- Если строка начинается или заканчивается символом подчеркивания (_), возникает ошибка.
- Если между знаком минуса (-) или плюса (+) в начале строки и числом расположен символ подчеркивания (_), возникает ошибка

Имя команды определяется типом данных *Out* (результат преобразования). Например, если *Out* относится к типу данных BYTE, команда будет иметь имя STRING_TO_BYTE.

Ниже показан пример для команды STRING_TO_BYTE, в котором *In* = « AB». Символы пробела в начале строки игнорируются.



Допустимый диапазон

Диапазон допустимых значений *In* зависит от типа данных *Out*, что показано ниже:

Тип данных *Out*	Диапазон допустимых значений *In* (максимальное количество байтов)*1
BYTE	3 байта (2 однобайтовых буквенно-цифровых символа + последний символ NULL)
WORD	5 байтов (4 однобайтовых буквенно-цифровых символа + последний символ NULL)
DWORD	9 байтов (8 однобайтовых буквенно-цифровых символов + последний символ NULL)
LWORD	17 байтов (16 однобайтовых буквенно-цифровых символов + последний символ NULL)

*1. Символы пробела ( ) и нули в начале текстовой строки, а также символы подчеркивания (_) в текстовой строке не включаются в число байтов.

Дополнительная информация

- Чтобы обрабатывать число со знаком как текстовую строку, используйте команду *STRING_TO_** *(Группа преобразования текстовых строк в целые числа)* на стр. 2-344.
- Для преобразования битовой строки в текстовую строку используйте команду ***_TO_STRING *(Группа преобразования битовых строк в текстовые строки)* на стр. 2-325.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типу данных переменной *Out*.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Текстовая строка в *In* не выражает число.

b) Результат преобразования выходит за диапазон значений, допускаемых типом данных переменной *Out*.

STRING_TO_** (Группа преобразования текстовых строк в вещественные числа)

Эти команды преобразуют текстовые строки в вещественные значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
STRING_TO_**	Группа преобразования текстовых строк в вещественные числа	FUN	(@)STRING_TO_** EN ENO In Out "**" — вещественный тип данных	Out:=STRING_TO_** (In); "**" — вещественный тип данных

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Макс. 311 байт (310 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	"
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Out														OK	OK					

Функция

Эти команды преобразуют текстовую строку *In* в вещественное значение.
Имя команды определяется типом данных *Out* (результат преобразования). Например, если *Out* относится к типу данных LREAL, команда будет иметь имя STRING_TO_LREAL.

Формат текстовой строки в *In* приведен ниже.

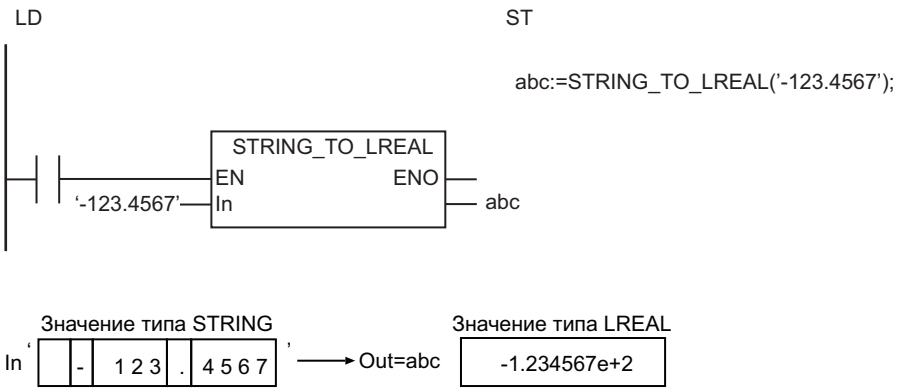
Имя	Формат
Знак	• Идущие подряд символы пробела, расположенные в начале текстовой строки, игнорируются. Если за ними следует одиночный знак плюса (+) или минуса (-), он воспринимается как знак. • Знак плюса (+) можно опускать. • Идущие подряд символы пробела, расположенные после знака, игнорируются.
Целая часть	• Символы после знака и до десятичной точки принимаются за целую часть. Идущие подряд символы пробела, расположенные после знака, в целую часть не включаются. Знак иногда может быть опущен. • Если десятичная точка и дробная часть опущены, то за целую часть принимаются символы вплоть до показателя степени. • Если опущены десятичная точка, дробная часть и показатель степени, то за целую часть принимаются символы вплоть до конца текстовой строки. • Целая часть состоит из символов от «0» до «9». • Целая часть не может быть опущена. • Максимальное количество цифр в целой части определяется как максимальная длина текстовой строки (1985) минус общее количество байтов, занимаемое следующим: знак, десятичная точка, дробная часть, показатель степени и пробелы до и после знака.
Десятичная точка	• Одиночная точка (.), следующая за целой частью, принимается за десятичную точку. • При отсутствии дробной части десятичную точку следует опустить.
Дробная часть	• Символы после десятичной точки и до показателя степени считаются дробной частью. • Если показатель степени опущен, за дробную часть принимаются символы вплоть до конца текстовой строки. • Дробная часть состоит из символов от «0» до «9». • Дробная часть может быть опущена. • Дробная часть может состоять максимум из 15 цифр. • При отсутствии десятичной точки также отсутствует и дробная часть.
Показатель степени	• Показатель степени состоит из одного символа «e» или «E» после дробной части, следующего за ним одного знака «плюс» (+) или «минус» (-) и всех оставшихся символов до конца текстовой строки. • При отсутствии дробной части за показатель степени принимается описанная выше текстовая строка, расположенная после десятичной точки. • Если в строке нет ни дробной части, ни десятичной точки, за показатель степени принимается описанная выше текстовая строка, следующая за целой частью. • Числовая часть показателя степени состоит из символов от «0» до «9». • Показатель степени можно опускать. • Числовая часть показателя степени может состоять максимум из трех цифр.

Если значение *In* равно «+inf», то значение *Out* равно положительной бесконечности. Если значение *In* равно «-inf», то значение *Out* равно отрицательной бесконечности. В обоих случаях символы не чувствительны к регистру.

Пример записи

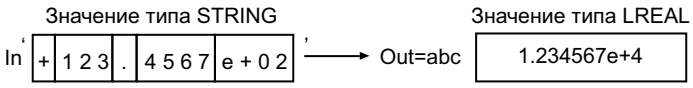
● Пример 1.

В следующем примере используются знак, десятичная точка и дробная часть, а показатель степени не используется.



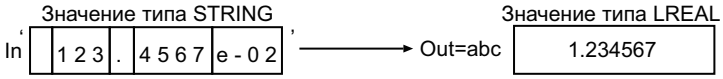
● Пример 2.

В следующем примере используются знак, десятичная точка, дробная часть и показатель степени.



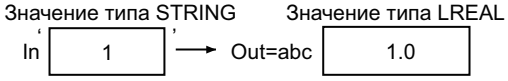
● Пример 3.

В следующем примере не используется знак, но используются десятичная точка, дробная часть и показатель степени.



● Пример 4.

В следующем примере не используются знак, дробная часть, десятичная точка и показатель степени.



Дополнительная информация

Для преобразования вещественного числа в текстовую строку используйте команду ***_TO_STRING (Группа преобразования вещественных чисел в текстовые строки)* на стр. 2-327.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типу данных переменной *Out*.
- Одиночный символ подчеркивания (_), расположенный в любом месте строки в *In*, игнорируется.

- Если строка в *In* начинается или заканчивается символом подчеркивания (_), возникает ошибка.
- Если в каком-либо месте строки в *In* имеются два или больше символов подчеркивания (_) подряд, возникает ошибка.
- Если между знаком минуса (-) или плюса (+) и числом в строке *In* расположен символ подчеркивания (_), возникает ошибка
- Если точность значения *In* превышает точность типа данных *Out*, значение округляется.
- Если значение *In* ближе к 0, чем минимальное значение типа данных *Out*, значение *Out* будет равно 0.
- Если значение *In* выходит за диапазон допустимых значений *Out*, *Out* будет положительной бесконечностью в случае положительного числа или отрицательной бесконечностью в случае отрицательного числа.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Текстовая строка в *In* не выражает число.
  b) В текстовой строке в *In* есть десятичная точка, но нет дробной части.

TO_** (Группа преобразования в целые числа)

Эти команды преобразуют целочисленные значения, битовые строки, вещественные значения и текстовые строки в целочисленные значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TO_**	Группа преобразования в целые числа	FUN	(@)TO_** EN ENO In Out "**" — целочисленный тип данных.	Out:=TO_**(In); "**" — целочисленный тип данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	*2
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-355.

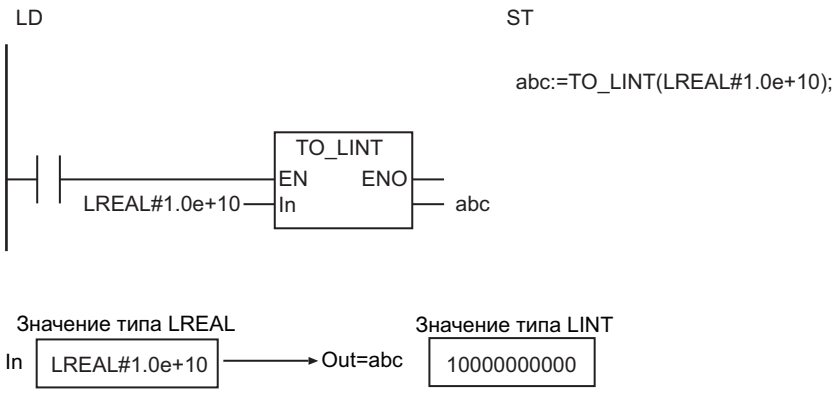
*2. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					OK
Out						OK	OK	OK	OK	OK	OK	OK	OK							

Функция

Эти команды преобразуют целочисленное значение, битовую строку, вещественное значение или текстовую строку *In* в целочисленное значение.
Имя команды определяется типом данных *Out* (результат преобразования). Например, если *Out* относится к типу данных LINT, команда будет иметь имя TO_LINT.

Ниже показан пример для команды TO_LINT, в котором *In* = LREAL#1.0e+10.



- Преобразование выполняется с точностью, соответствующей числу значащих разрядов типа данных *In*. Если *In* является вещественным числом, дробная часть значения отбрасывается, значение округляется до ближайшего целого числа.

Округление

В следующей таблице показано, как округляются значения.

Значение дробной части	Описание	Примеры
Меньше 0,5	Дробная часть отбрасывается.	1,49 → 1
0,5	Если разряд единиц содержит четное число, дробная часть отбрасывается. Если это нечетное число, то значение округляется в большую сторону.	1,50 → 2 2,50 → 2
Больше 0,5	Дробная часть отбрасывается, число округляется в большую сторону.	1,51 → 2

Допустимый диапазон

Диапазоны допустимых значений для *In* и *Out* зависят от их типов данных. Сведения о диапазоне допустимых значений для каждого типа данных см. в соответствующем разделе: *Допустимый диапазон* на стр. 2-300 для **_TO_*** (Группа преобразования целых чисел в целые числа), *Допустимый диапазон* на стр. 2-309 для **_TO_*** (Группа преобразования битовых строк в целые числа) и *Допустимый диапазон* на стр. 2-316 для **_TO_*** (Группа преобразования вещественных чисел в целые числа).

Сведения о характеристиках для случая, когда *In* является строковым типом (STRING), см. в разделе *Допустимый диапазон* на стр. 2-345 для STRING_TO_** (Группа преобразования текстовых строк в целые числа).

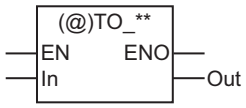
Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типу данных переменной *Out*.
- Если тип данных *In* соответствует битовой строке и размеры типов данных *In* и *Out* различаются, значение обрабатывается следующим образом.
  a) Если размер (разрядность) значений *Out* превышает размер (разрядность) значений *In*, старшие разряды *Out* будут содержать 0.
  b) Если разрядность *Out* меньше разрядности *In*, то старшие разряды будут отброшены.

- Если *In* является строковым типом (STRING), соблюдайте указанные ниже меры предосторожности.
  a) Если первым символом в *In* является знак минуса (-) или знак плюса (+), он обрабатывается как знак.
  b) За исключением знака минуса (-) или плюса (+) в начале строки, строка *In* должна представлять собой последовательность символов от «0» до «9». В текстовой строке допускается наличие символов подчеркивания (_) и пробелов до или после знака (-) или (+).
- Если результат преобразования выйдет за диапазон допустимых значений *Out*, *Out* будет содержать неопределенное значение. Всегда следите за тем, чтобы значение *In* находилось в пределах допустимого диапазона, чтобы результат преобразования не вышел за диапазон допустимых значений *Out*.
- В указанных ниже случаях происходит ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) *In* является строковым типом (STRING), но текстовая строка не выражает число.

TO_** (Группа преобразования в битовые строки)

Эти команды преобразуют целочисленные значения, битовые строки, вещественные значения и текстовые строки в битовые строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TO_**	Группа преобразования в битовые строки	FUN	 "**" — тип данных, относящийся к битовым строкам.	Out:=TO_**(In); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1	---	*2
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-358.

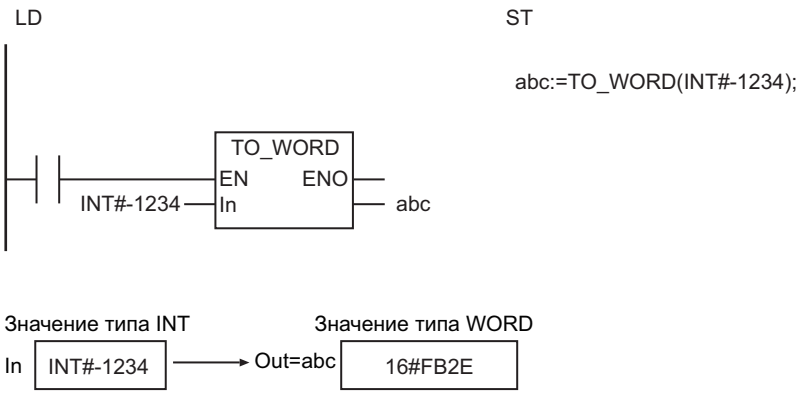
*2. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					OK
Out		OK	OK	OK	OK															

Функция

Эти команды преобразуют целочисленное значение, битовую строку, вещественное значение или текстовую строку *In* в битовую строку.
Имя команды определяется типом данных *Out* (результат преобразования). Например, если *Out* относится к типу данных WORD, команда будет иметь имя TO_WORD.

Ниже показан пример для команды TO_WORD, в котором *In* = INT#-1234.



Допустимый диапазон

Диапазоны допустимых значений для *In* и *Out* зависят от их типов данных. Сведения о диапазоне допустимых значений для каждого типа данных см. в соответствующем разделе: *Допустимый диапазон* на стр. 2-303 для **_TO_*** (Группа преобразования целых чисел в битовые строки) и *Допустимый диапазон* на стр. 2-312 для **_TO_*** (Группа преобразования битовых строк в битовые строки).

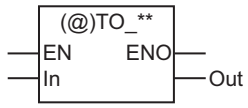
Сведения о характеристиках для случая, когда *In* является строковым типом (STRING), см. в разделе *Допустимый диапазон* на стр. 2-348 для STRING_TO_** (Группа преобразования текстовых строк в битовые строки).

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типу данных переменной *Out*.
- Если результат преобразования выйдет за диапазон допустимых значений *Out*, *Out* будет содержать неопределенное значение. Всегда следите за тем, чтобы значение *In* находилось в пределах допустимого диапазона, чтобы результат преобразования не вышел за диапазон допустимых значений *Out*.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) *In* является строковым типом (STRING), но текстовая строка не выражает число.

TO_** (Группа преобразования в вещественные числа)

Эти команды преобразуют целочисленные значения, битовые строки, вещественные значения и текстовые строки в вещественные значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TO_**	Группа преобразования в вещественные числа	FUN	 "**" — вещественный тип данных	Out:=TO_**(In); "**" — вещественный тип данных

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	*1 *2	---	*3
Out	Результат преобразования	Выход	Результат преобразования	*1	---	---

*1. Диапазоны допустимых значений зависят от типов данных *In* и *Out*. Дополнительные сведения см. в разделе *Допустимый диапазон* на стр. 2-360.

*2. Для строковых данных (STRING) диапазон допустимых значений составляет макс. 311 байт (310 однобайтовых буквенно-цифровых символов + последний символ NULL).

*3. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					OK
Out														OK	OK					

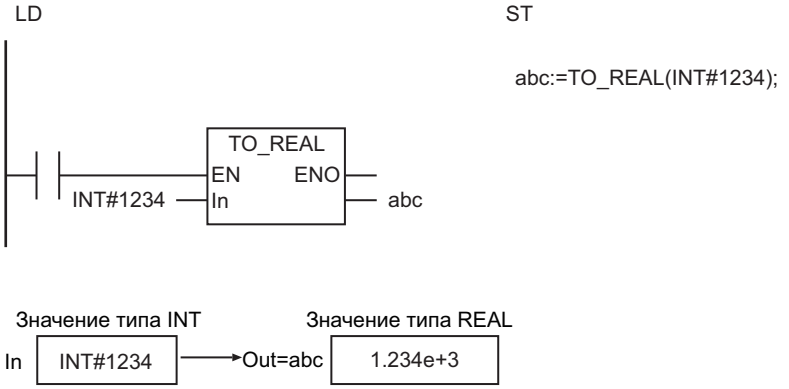
Функция

Эти команды преобразуют целочисленное значение, битовую строку, вещественное значение или текстовую строку *In* в вещественное значение.

Имя команды определяется типом данных *Out* (результат преобразования). Например, если *Out* относится к типу данных REAL, команда будет иметь имя TO_REAL.

Если значение *In* является положительной или отрицательной бесконечностью, значение *Out* также является положительной или отрицательной бесконечностью.

Ниже показан пример для команды TO_REAL, в котором *In* = INT#1234.



Допустимый диапазон

Диапазоны допустимых значений для *In* и *Out* зависят от их типов данных. Сведения о диапазоне допустимых значений для каждого типа данных см. в соответствующем разделе: *Допустимый диапазон* на стр. 2-306 для **_TO_*** (Группа преобразования целых чисел в вещественные числа), *Допустимый диапазон* на стр. 2-314 для **_TO_*** (Группа преобразования битовых строк в вещественные числа) и *Допустимый диапазон* на стр. 2-322 для **_TO_*** (Группа преобразования вещественных чисел в вещественные числа).

Сведения о характеристиках для случая, когда *In* является строковым типом (STRING), см. в разделе *Функция* на стр. 2-350 для STRING_TO_** (Группа преобразования текстовых строк в вещественные числа).

Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте правильное имя команды, соответствующее типу данных переменной *Out*.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) *In* является строковым типом (STRING), но текстовая строка не выражает число.

EnumToNum

Команда EnumToNum преобразует значение перечислимого типа в значение типа DINT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EnumToNum	Перечисление в целое число	FUN	(@)EnumToNum EN ENO In Out	Out:=EnumToNum(In);

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	---	---	0
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	Перечисление																			
Out												OK								

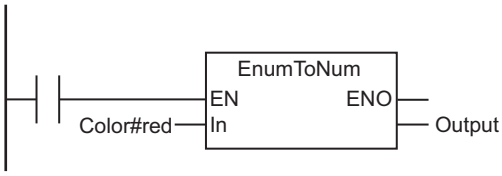
Функция

Команда EnumToNum преобразует значение *In* (данные для преобразования), являющееся перечислением, в значение типа DINT и выводит результат в переменную *Out* (результат преобразования).

С помощью этой команды можно, например, отображать значения переменной перечислимого типа на экране панели оператора или другого устройства отображения, не поддерживающего работу с переменными перечислимого типа.

Показанный ниже пример демонстрирует, как команда преобразует перечислитель *red* перечисления *Color* в значение и выводит это значение в переменную *Output* типа DINT.
Если значение перечислителя *red* равно *0*, то значение *Output* будет равно DINT#0.

LD



ST

```
Output:=EnumToNum(Color#red);
```

Пример программы

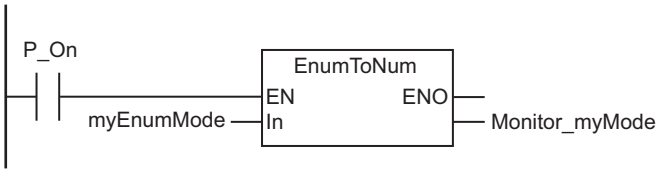
В этом примере с помощью перечислимого типа данных EnumMode определяется режим работы пользовательской программы.
Для мониторинга режима работы на терминале HMI преобразуется значение переменной *myEnumMode* (перечисление с типом данных EnumMode). Преобразованное значение выводится в переменную *Monitor_myMode* типа DINT. Например, если значение *myEnumMode* равно «mode2», то значение *Monitor_myMode* будет равно 2.

Определение типа данных

Имя	Значение перечисления	Комментарий
EnumMode	---	Перечислимый тип данных
mode0	0	Член перечисления
mode1	1	Член перечисления
mode2	2	Член перечисления

Программа на языке LD

Имя	Тип данных	По умолчанию	Комментарий
myEnumMode	EnumMode	mode0	Значение режима с перечислимым типом данных
Monitor_myMode	DINT	0	Контролируемое значение режима



Программа на языке ST

Имя	Тип данных	По умолчанию	Комментарий
myEnumMode	EnumMode	mode0	Значение режима с перечислимым типом данных
Monitor_myMode	DINT	0	Контролируемое значение режима

```
Monitor_myMode:=EnumToNum(myEnumMode);
```

NumToEnum

Команда NumToEnum преобразует значение типа DINT в значение перечислимого типа.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NumToEnum	Целое число в перечисление	FUN	(@)NumToEnum EN In InOut Out	NumToEnum(In, InOut);

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	0
InOut	Результат преобразования	Вход-выход	Результат преобразования	---	---	---
Out	Возвращаемое значение	Выход	ИСТИНА: команда была выполнена нормально. ЛОЖЬ: команда не была выполнена или произошла ошибка.	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In												ОК								
InOut	Перечисление																			
Out	ОК																			

Функция

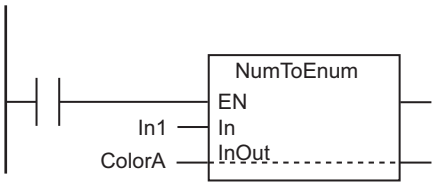
Команда NumToEnum преобразует значение *In* (данные для преобразования) типа DINT в значение перечислимого типа и выводит это значение в переменную *InOut* (результат преобразования).

С помощью этой команды можно, например, изменить значение переменной перечислимого типа с панели оператора или другого устройства отображения, не поддерживающего переменные перечислимого типа.

Показанный ниже пример демонстрирует, как команда преобразует значение переменной *In1* типа DINT и выводит результаты в переменную *ColorA*, которая имеет перечислимый тип данных *Color*.

Если значению 1 перечисления типа *Color* соответствует перечислитель *green*, а значение *In1* равно *1*, то значение *ColorA* будет равно *green*.

LD



ST

```
NumToEnum(In1,ColorA);
```

Дополнительная информация

При использовании данной команды в лестничной диаграмме с помощью переменной *Out* можно проверить, не выходит ли значение *In* за диапазон допустимых значений *InOut*.

Меры предосторожности для обеспечения надлежащей эксплуатации

Если значение *In* выходит за пределы диапазона допустимых значений *InOut*, возникает ошибка. Выход *Out* будет содержать ЛОЖЬ, а значение *InOut* не изменится.

Пример программы

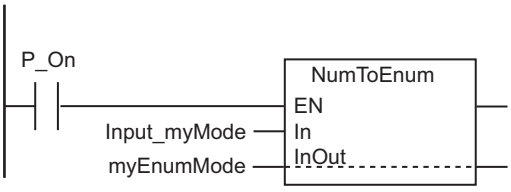
В этом примере с помощью перечислимого типа данных EnumMode определяется режим работы пользовательской программы.
Для изменения режима работы с терминала HMI записывается соответствующее значение в переменную *Input_myMode* типа DINT.
В пользовательской программе значение *Input_myMode* преобразуется, и преобразованное значение выводится в переменную *myEnumMode* (перечисление с типом данных EnumMode). Например, если значение *Input_myMode* равно *1*, то значение *myEnumMode* будет равно «mode1».

Определение типа данных

Имя	Значение перечисления	Комментарий
EnumMode	---	Перечислимый тип данных
mode0	0	Член перечисления
mode1	1	Член перечисления
mode2	2	Член перечисления

Программа на языке LD

Имя	Тип данных	По умолчанию	Комментарий
myEnumMode	EnumMode	mode0	Значение режима с перечислимым типом данных
Input_myMode	DINT	0	Значение режима, в который нужно перейти



Программа на языке ST

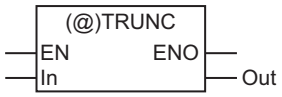
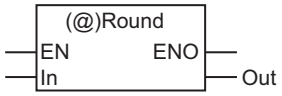
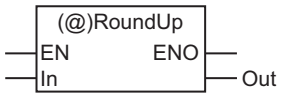
Имя	Тип данных	По умолчанию	Комментарий
myEnumMode	EnumMode	mode0	Значение режима с перечислимым типом данных
Input_myMode	DINT	0	Значение режима, в который нужно перейти

```
NumToEnum(Input_myMode, myEnumMode);
```

TRUNC, Round и RoundUp

Эти команды преобразуют вещественные значения в целочисленные значения.

TRUNC : Усекает вещественное число до целого.
Round : Округляет вещественное число до ближайшего целого числа в большую или меньшую сторону.
RoundUp : Округляет вещественное число до ближайшего целого числа в большую сторону.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TRUNC	Усечение	FUN		Out:=TRUNC(In);
Round	Округление вещественного числа	FUN		Out:=Round(In);
RoundUp	Округление вещественного числа к большему	FUN		Out:=RoundUp(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	*1
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														ОК	ОК					
Out												ОК	ОК							

Функция

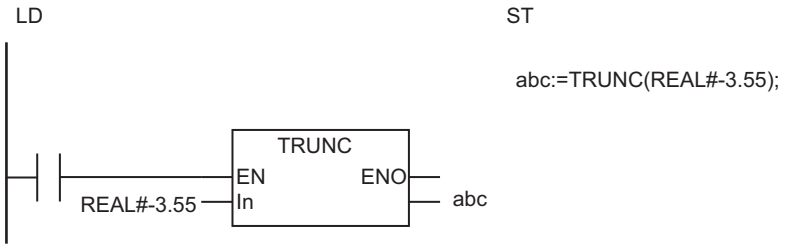
Эти команды делают вещественное значение в *In* целочисленным значением, отсекая от него дробную часть.

TRUNC

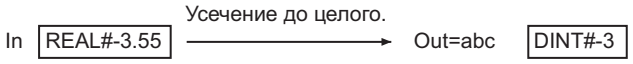
Команда TRUNC усекает число до разряда единиц.

Ниже показан пример для команды TRUNC, в котором *In* = REAL#-3.55.

Значение переменной *abc* будет равно DINT#-3.



Команда TRUNC усекает число до целого, отбрасывая дробную часть.
Значение ***In*** равно REAL#-3.55, поэтому значение ***abc*** будет равно DINT#-3.



Round

Команда Round округляет число до разряда единиц.

● Округление

В следующей таблице показано, как округляются значения.

Значение дробной части	Описание	Примеры
Меньше 0,5	Дробная часть отбрасывается.	1,49 → 1 -1,49 → -1
0,5	Если разряд единиц содержит четное число, дробная часть отбрасывается. Если это нечетное число, то значение округляется в большую сторону.	1,50 → 2 2,50 → 2 -1,50 → -2 -2,50 → -2
Больше 0,5	Дробная часть отбрасывается, число округляется в большую сторону.	1,51 → 2 -1,51 → -2

RoundUp

Команда RoundUp округляет число до разряда единиц в большую сторону.

Различия в работе

В следующей таблице перечислены различия в работе этих трех команд.

Входное значение	Выходное значение		
	TRUNC	Round	RoundUp
REAL#1.6	DINT#1	DINT#2	DINT#2

Входное значение	Выходное значение		
	TRUNC	Round	RoundUp
REAL#1.5	DINT#1	DINT#2	DINT#2
REAL#1.5	DINT#1	DINT#1	DINT#2
REAL#2.5	DINT#2	DINT#2	DINT#3
REAL#-1.6	DINT#-1	DINT#-2	DINT#-2
REAL#-1.5	DINT#-1	DINT#-2	DINT#-2
REAL#-1.4	DINT#-1	DINT#-1	DINT#-2
REAL#-2.5	DINT#-2	DINT#-2	DINT#-3

Дополнительная информация

Если *In* имеет тип данных REAL, то *Out* будет иметь тип данных DINT.
Если *In* имеет тип данных LREAL, то *Out* будет иметь тип данных LINT.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если результат преобразования выйдет за диапазон допустимых значений *Out*, *Out* будет содержать неопределенное значение.
  Всегда следите за тем, чтобы значение *In* находилось в пределах допустимого диапазона, чтобы результат преобразования не вышел за диапазон допустимых значений *Out*.
- Если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

Команды обработки битовых строк

Команда	Имя	Стр.
AND (&), OR и XOR	Логическое И/Логическое ИЛИ/Логическое исключающее ИЛИ	стр. 2-370
XORN	Логическое исключающее ИЛИ-НЕ	стр. 2-373
NOT	Побитовая инверсия	стр. 2-375
AryAnd, AryOr, AryXor и AryXorN	Логическое И массивов/Логическое ИЛИ массивов/Логическое исключающее ИЛИ массивов/Логическое исключающее ИЛИ-НЕ массивов	стр. 2-377

AND (&), OR и XOR

Эти команды выполняют перечисленные ниже побитовые операции над несколькими переменными логического типа (BOOL) или битовыми строками.

AND (&) : Логическое И
OR : Логическое ИЛИ
XOR : Логическое исключающее ИЛИ

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AND (&)	Логическое И	FUN	(@)AND EN ENO In1 Out : : InN (@)& EN ENO In1 Out : : InN	Out:=In1 AND ··AND InN; Out:=In1 & ··& InN;
OR	Логическое ИЛИ	FUN	(@)OR EN ENO In1 Out : : InN	Out:=In1 OR ··OR InN;
XOR	Логическое исключающее ИЛИ	FUN	(@)XOR EN ENO In1 Out : : InN	Out:=In1 XOR ··XOR InN;

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Данные для обработки	Вход	Данные для обработки N = 2...5	Зависит от типа данных.	---	0*1
Out	Результат обработки	Выход	Результат обработки	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3 и будут опущены входные параметры, подключаемые к *In1* и *In2*, то для них будут применены значения по умолчанию. Но если будет опущен входной параметр, подключаемый к *In3*, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN	OK	OK	OK	OK	OK															
Out	Тип данных должен быть таким же, как у *In1*...*InN*.																			

Функция

Эти команды выполняют побитовые операции с данными для обработки *In1*...*InN*, которые представляют собой несколько логических переменных (BOOL) или битовых строк.
Для переменных *In1*...*InN* и *Out* должен использоваться один и тот же тип данных.

При обработке трех или большего числа значений операции выполняются следующим образом.

1 Выполняется операция над *In1* и *In2*.

2 Выполняется операция над результатом шага 1 и *In3*.

3 Выполняется операция над результатом шага 2 и *In4* .

Далее операции выполняются так, как показано выше.

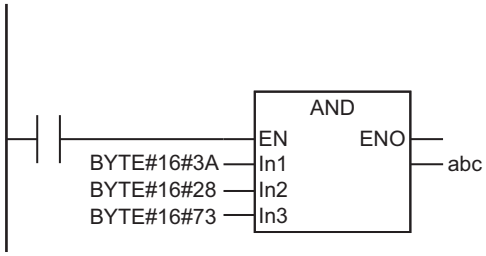
AND (&)

Если оба бита находятся в состоянии ИСТИНА, результат обработки равен ИСТИНА. В противном случае результат обработки равен ЛОЖЬ.

Бит *In1*	Бит *In2*	Бит *Out*
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
ЛОЖЬ	ИСТИНА	ЛОЖЬ
ИСТИНА	ЛОЖЬ	ЛОЖЬ
ИСТИНА	ИСТИНА	ИСТИНА

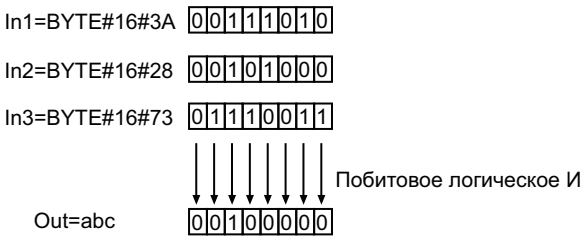
В представленном ниже примере для команды AND переменные *In1*, *In2* и *In3* имеют значения BYTE#16#3A, BYTE#16#28 и BYTE#16#73 соответственно.

LD



ST

```
abc:=BYTE#16#3A AND BYTE#16#28 AND BYTE#16#73;
```



Команды AND и & работают абсолютно одинаково, отличаясь лишь формой записи. Используйте ту форму записи, которая вам более удобна.

OR

Если оба бита находятся в состоянии ЛОЖЬ, результат обработки равен ЛОЖЬ. В противном случае результат обработки равен ИСТИНА.

Бит *In1*	Бит *In2*	Бит *Out*
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
ЛОЖЬ	ИСТИНА	ИСТИНА
ИСТИНА	ЛОЖЬ	ИСТИНА
ИСТИНА	ИСТИНА	ИСТИНА

XOR

Если оба бита находятся в одинаковом состоянии, результат обработки равен ЛОЖЬ. Если один бит находится в состоянии ИСТИНА, а другой — в состоянии ЛОЖЬ, результат обработки равен ИСТИНА.

Бит *In1*	Бит *In2*	Бит *Out*
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
ЛОЖЬ	ИСТИНА	ИСТИНА
ИСТИНА	ЛОЖЬ	ИСТИНА
ИСТИНА	ИСТИНА	ЛОЖЬ

Дополнительная информация

При использовании показанной ниже формы записи в программе на языке ST количество входных переменных ничем не ограничено.

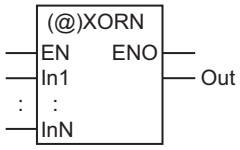
```
Out:=In1 AND In2 AND In3 AND In4 AND In5 AND In6 ···
Out:=In1 & In2 & In3 & In4 & In5 & In6 ···
Out:=In1 OR In2 OR In3 OR In4 OR In5 OR In6 ···
Out:=In1 XOR In2 XOR In3 XOR In4 XOR In5 XOR In6 ···
```

Меры предосторожности для обеспечения надлежащей эксплуатации

Для переменных *In1*...*InN* и *Out* должен использоваться один и тот же тип данных.
В противном случае произойдет ошибка сборки.

XORN

Команда XORN выполняет побитовую операцию «логическое исключающее ИЛИ-НЕ» над несколькими переменными логического типа (BOOL) или битовыми строками.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
XORN	Логическое исключающее ИЛИ-НЕ	FUN		Out:=In1 XOR NOT ·· XOR NOT InN;

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Данные для обработки	Вход	Данные для обработки N = 2...5	Зависит от типа данных.	---	0*1
Out	Результат обработки	Выход	Результат обработки	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3 и будут опущены входные параметры, подключаемые к *In1* и *In2*, то для них будут применены значения по умолчанию. Но если будет опущен входной параметр, подключаемый к *In3*, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN	ОК	ОК	ОК	ОК	ОК															
Out	Тип данных должен быть таким же, как у *In1*...*InN*.																			

Функция

Эти команды выполняют побитовые операции с данными для обработки *In1*...*InN*, которые представляют собой несколько логических переменных (BOOL) или битовых строк.
Для переменных *In1*...*InN* и *Out* должен использоваться один и тот же тип данных.

При обработке трех или большего числа значений операции выполняются следующим образом.

1 Выполняется операция над *In1* и *In2*.

2 Выполняется операция над результатом шага 1 и *In3*.

3 Выполняется операция над результатом шага 2 и *In4* .

Далее операции выполняются так, как показано выше.

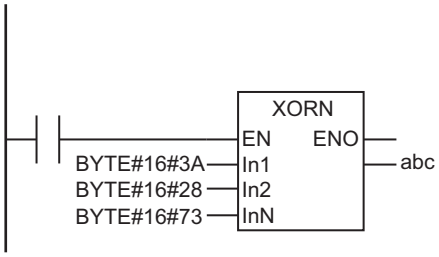
XORN

Эта команда работает в соответствии с таблицей ниже.
Если оба бита находятся в одинаковом состоянии, результат обработки равен ИСТИНА. В противном случае результат обработки равен ЛОЖЬ.

Бит *In1*	Бит *In2*	Бит *Out*
ЛОЖЬ	ЛОЖЬ	ИСТИНА
ЛОЖЬ	ИСТИНА	ЛОЖЬ
ИСТИНА	ЛОЖЬ	ЛОЖЬ
ИСТИНА	ИСТИНА	ИСТИНА

Ниже показан пример, в котором переменные *In1*, *In2* и *In3* имеют значения BYTE#16#3A, BYTE#16#28 и BYTE#16#73 соответственно.

LD



ST

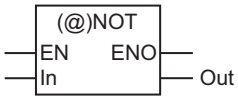
```
abc:=BYTE#16#3A XOR NOT BYTE#16#28
      XOR NOT BYTE#16#73;
```

Меры предосторожности для обеспечения надлежащей эксплуатации

Для переменных *In1*...*InN* и *Out* должен использоваться один и тот же тип данных.
В противном случае произойдет ошибка сборки.

NOT

Команда NOT инвертирует каждый бит переменной логического типа (BOOL) или битовой строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NOT	Побитовая инверсия	FUN		Out:=NOT(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для обработки	Вход	Данные для обработки	Зависит от типа данных.	---	*1
Out	Результат обработки	Выход	Результат обработки	Зависит от типа данных.	---	---

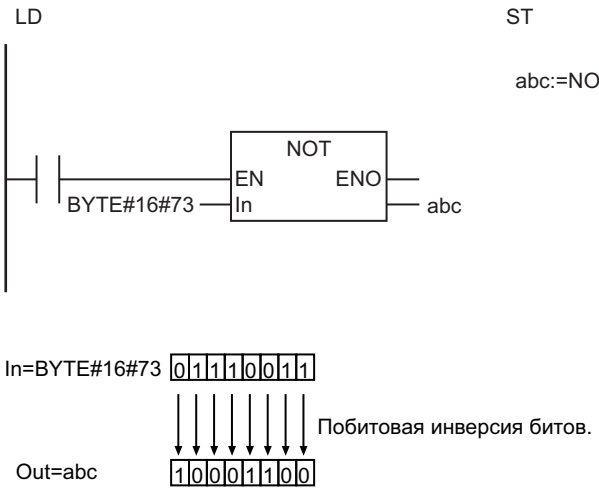
*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	OK	OK	OK	OK	OK															
Out	Тип данных должен быть таким же, как у *In1*.																			

Функция

Команда NOT инвертирует биты в переменной *In* (данные для обработки), содержащей логическое значение (BOOL) или битовую строку.
In и *Out* (результат обработки) должны иметь одинаковое количество битов, т. е. должны быть одного типа данных.

В показанном ниже примере программы *In* = BYTE#16#73.



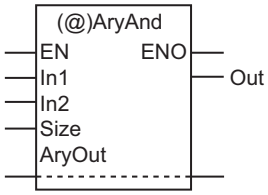
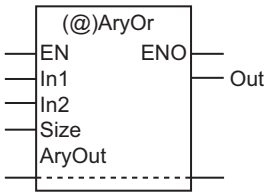
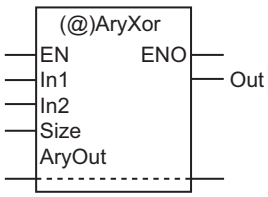
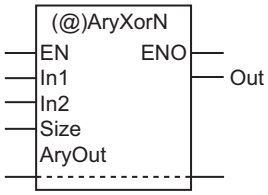
Меры предосторожности для обеспечения надлежащей эксплуатации

Типы данных *In* и *Out* должны быть одинаковыми.
В противном случае произойдет ошибка сборки.

AryAnd, AryOr, AryXor и AryXorN

Эти команды выполняют перечисленные ниже побитовые операции над каждой парой элементов двух массивов, содержащих значения логического типа или битовые строки.

AryAnd : Логическое И
AryOr : Логическое ИЛИ
AryXor : Логическое исключающее ИЛИ
AryXorN : Логическое исключающее ИЛИ-НЕ

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryAnd	Логическое И массивов	FUN		AryAnd(In1, In2, Size, AryOut);
AryOr	Логическое ИЛИ массивов	FUN		AryOr(In1, In2, Size, AryOut);
AryXor	Логическое исключающее ИЛИ массивов	FUN		AryXor(In1, In2, Size, AryOut);
AryXorN	Логическое исключающее ИЛИ-НЕ массивов	FUN		AryXorN(In1, In2, Size, AryOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1[] и In2[] (массивы)	Массив для обработки	Вход	Массив для обработки	Зависит от типа данных.	---	*1
Size	Количество элементов		Количество элементов для обработки			1
AryOut[] (массив)	Массив с результатами обработки	Вход-выход	Массив с результатами обработки	Зависит от типа данных.	---	---

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[] (массив)	ОК	ОК	ОК	ОК	ОК															
In2[] (массив)	Тип данных должен быть таким же, как у In1[].																			
Size							ОК													
AryOut[] (массив)	Тип данных должен быть таким же, как у In1[].																			
Out	ОК																			

Функция

Эти команды выполняют побитовые операции над указанным количеством элементов (*Size*) массивов In1[] и In2[], начиная с начала массивов. Результаты операций записываются в соответствующие элементы массива AryOut[].
Массивы In1[], In2[] и AryOut[] должны быть одного типа данных.

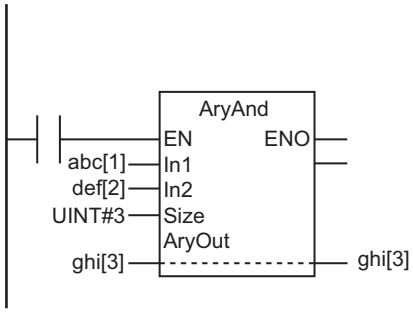
AryAnd

Если оба бита находятся в состоянии ИСТИНА, то результат обработки равен ИСТИНА. В противном случае результат обработки равен ЛОЖЬ.

Бит элемента в In1[]	Бит элемента в In2[]	Бит AryOut[]
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
ЛОЖЬ	ИСТИНА	ЛОЖЬ
ИСТИНА	ЛОЖЬ	ЛОЖЬ
ИСТИНА	ИСТИНА	ИСТИНА

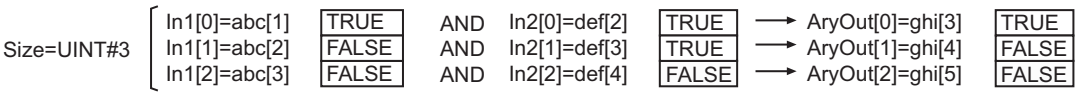
Ниже показан пример использования команды AryAnd для случая, когда *Size* = UINT#3.

LD



ST

```
AryAnd(abc[1], def[2], UINT#3, ghi[3]);
```



AryOr

Если оба бита находятся в состоянии ЛОЖЬ, то результат обработки равен ЛОЖЬ. В противном случае результат обработки равен ИСТИНА.

Бит элемента в In1[]	Бит элемента в In2[]	Бит AryOut[]
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
ЛОЖЬ	ИСТИНА	ИСТИНА
ИСТИНА	ЛОЖЬ	ИСТИНА
ИСТИНА	ИСТИНА	ИСТИНА

AryXor

Если оба бита находятся в одинаковом состоянии, то результат обработки равен ЛОЖЬ. Если один бит находится в состоянии ИСТИНА, а другой — в состоянии ЛОЖЬ, то результат обработки равен ИСТИНА.

Бит элемента в In1[]	Бит элемента в In2[]	Бит AryOut[]
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
ЛОЖЬ	ИСТИНА	ИСТИНА
ИСТИНА	ЛОЖЬ	ИСТИНА
ИСТИНА	ИСТИНА	ЛОЖЬ

AryXorN

Если оба бита находятся в одинаковом состоянии, результат обработки равен ИСТИНА. Если один бит находится в состоянии ИСТИНА, а другой — в состоянии ЛОЖЬ, то результат обработки равен ЛОЖЬ.

Бит элемента в In1[]	Бит элемента в In2[]	Бит AryOut[]
ЛОЖЬ	ЛОЖЬ	ИСТИНА
ЛОЖЬ	ИСТИНА	ЛОЖЬ
ИСТИНА	ЛОЖЬ	ЛОЖЬ
ИСТИНА	ИСТИНА	ИСТИНА

Меры предосторожности для обеспечения надлежащей эксплуатации

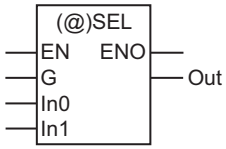
- Типы данных In1[], In2[] и AryOut[] должны быть одинаковыми.
  При использовании разных типов данных произойдет ошибка сборки.
- Количество элементов в массиве AryOut[] должно быть равно или больше значения *Size*.
- Если *Size* = 0, значения в массиве AryOut[] не изменяются.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение *Size* превышает количество элементов массива In1[], In2[] или AryOut[].

Команды выбора

Команда	Имя	Стр.
SEL	Двоичный выбор	стр. 2-382
MUX	Мультиплексор	стр. 2-385
LIMIT	Ограничитель	стр. 2-388
Band	Зона нечувствительности	стр. 2-390
Zone	Мертвая зона	стр. 2-393
MAX и MIN	Максимум/Минимум	стр. 2-396
AryMax и AryMin	Максимум массива/Минимум массива	стр. 2-399
ArySearch	Поиск по массиву	стр. 2-402

SEL

Команда SEL выбирает один из двух вариантов.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SEL	Двоичный выбор	FUN		Out:=SEL(G, In0, In1);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
G	Выбор	Вход	ЛОЖЬ: выбрано *In0* ИСТИНА: выбрано *In1*	Зависит от типа данных.	---	ЛОЖЬ
In0 и In1	Варианты для выбора		Варианты для выбора			*1
Out	Результат выбора	Выход	Результат выбора	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
G	ОК																			
In0 и In1	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также допускается указывать перечисления.*1																			
Out	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также допускается указывать перечисления.*1																			

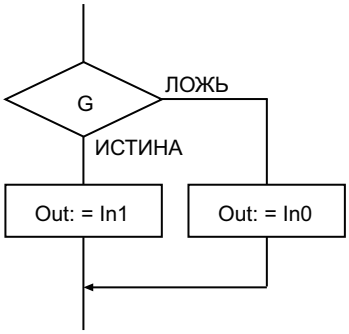
*1. Чтобы можно было указывать перечисления, требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Функция

Команда SEL делает выбор между двумя возможными вариантами: *In0* и *In1* (варианты для выбора).

Выбор между *In0* и *In1* определяется сигналом выбора *G*.

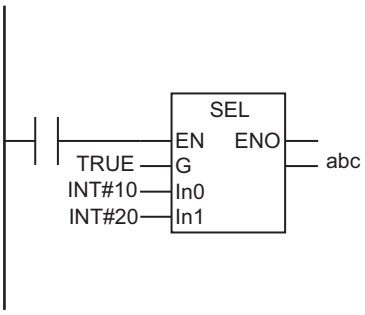
Если *G* = ЛОЖЬ, переменной *Out* присваивается значение *In0*. Если ИСТИНА, то присваивается значение *In1*.



Ниже показан пример, в котором переменные *In0*, *In1* и *G* имеют значения INT#10, INT#20 и ИСТИНА соответственно.

Значение переменной *abc* будет равно INT#20.

LD



ST

```
abc:=SEL(TRUE, INT#10, INT#20);
```

Команда SEL делает выбор между значениями ***In0*** и ***In1***.
G = ИСТИНА, поэтому выбирается значение ***In1*** (INT#20), и оно присваивается ***abc***.

G = ИСТИНА, поэтому ***In1*** присваивается ***Out***.

Дополнительная информация

Информация о версии

Если используются модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше, можно использовать команду *MUX* на стр. 2-385. Команда MUX производит выбор из нескольких (от двух до пяти) вариантов.

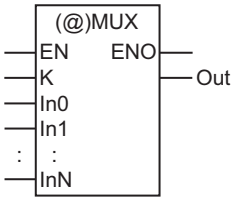
Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных переменных *In0*, *In1* и *Out* могут различаться, однако необходимо соблюдать следующие меры предосторожности.
  a) Диапазон допустимых значений *Out* должен включать в себя диапазоны допустимых значений *In0* и *In1*.

b) Типы данных переменных *In0*, *In1* и *Out* должны относиться к одной категории типов данных (т. е. они не должны относиться к разным категориям, таким как битовая строка и целое число или целое число и текстовая строка).

MUX

Команда MUX производит выбор из нескольких (от двух до пяти) вариантов.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MUX	Мультиплексор	FUN		Out:=MUX(K, In0, In1, ... , InN);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
K	Переключатель	Вход	0: выбрано *In0* 1: выбрано *In1* 2: выбрано *In2* 3: выбрано *In3* 4: выбрано *In4*	0...N	---	*1
In0...In1	Варианты для выбора		Варианты для выбора N = 1...4.*2	Зависит от типа данных.		0*3
Out	Результат выбора	Выход	Результат выбора	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

*2. В случае модуля ЦПУ с версией модуля 1.01 или более ранней и Sysmac Studio версии 1.02 или ниже N = 2...4.

*3. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 2 и будут опущены входные параметры, подключаемые к *In0* и *In1*, то для них будут применены значения по умолчанию. Но если будет опущен входной параметр, подключаемый к *In2*, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
K						OK *1			OK *1											
In0...InN	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также допускается указывать перечисления.*2																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также допускается указывать перечисления.*2																			

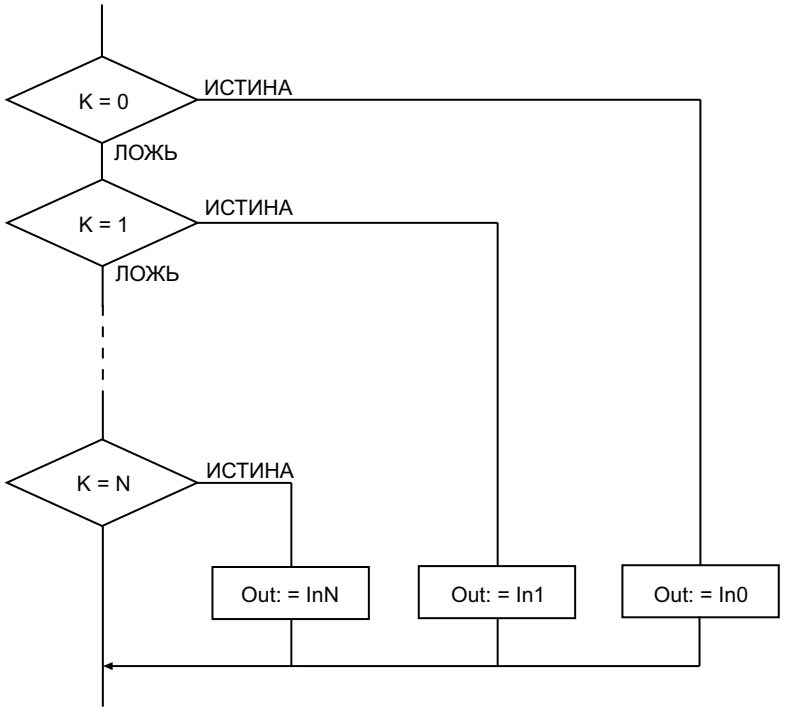
*1. Если используются модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше, используйте переменную типа ULINT.
В случае модуля ЦПУ с версией модуля 1.01 или более ранней и Sysmac Studio версии 1.02 или ниже используйте переменную типа USINT.

*2. Чтобы можно было указывать перечисления, требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Функция

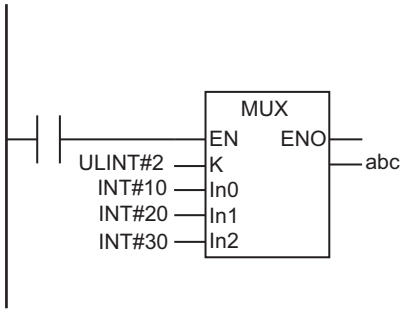
Команда MUX производит выбор из нескольких (от двух до пяти) вариантов: *In0...InN* (варианты для выбора).

Выбор между *In0...InN* определяется значением переключателя *K*.
В зависимости от значения *K* переменной *Out* присваивается то или иное значение. Если *K* = 0, присваивается значение *In0*. Если 1, то присваивается значение *In1* и т. д.



Ниже показан пример, в котором переменные *In0*, *In1*, *In2* и *K* имеют значения INT#10, INT#20, INT#30 и ULINT#2 соответственно. Значение переменной *abc* будет равно INT#30.

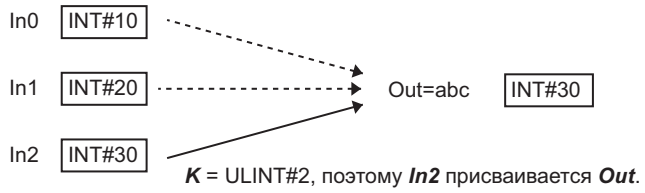
LD



ST

```
abc:=MUX(ULINT#2, INT#10, INT#20, INT#30);
```

Команда MUX делает выбор между значениями ***In0...InN***.
K = ULINT#2, поэтому выбирается значение ***In2*** (INT#30), и оно присваивается ***abc***.

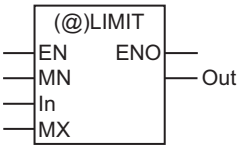


Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных переменных *In0...InN* и *Out* могут различаться, однако необходимо соблюдать следующие меры предосторожности.
  a) Диапазон допустимых значений *Out* должен включать в себя диапазоны допустимых значений *In0...InN*.
  b) Типы данных переменных *In0...InN* и *Out* должны относиться к одной категории типов данных (т. е. они не должны относиться к разным категориям, таким как битовая строка и целое число или целое число и текстовая строка).
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *K* находится за пределами допустимого диапазона (т. е. меньше 0 или больше N).

LIMIT

Команда LIMIT ограничивает значение входной переменной между указанными минимальным и максимальным значениями.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LIMIT	Ограничитель	FUN		Out:=LIMIT(MN, In, MX);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
MN	Минимальное значение	Вход	Минимальное значение ограничителя	Зависит от типа данных.	---	*1
In	Данные для ограничения		Данные для ограничения			
MX	Максимальное значение		Максимальное значение ограничителя			
Out	Результат обработки	Выход	Результат обработки	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MN						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
In						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
MX						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
Out						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					

Функция

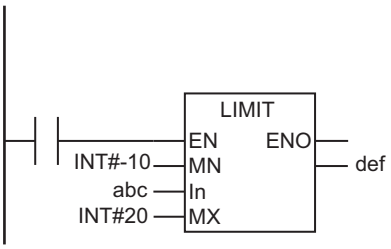
Команда LIMIT ограничивает значение *In* (данные для ограничения) между значением *MX* (максимальное значение) и значением *MN* (минимальное значение).
В таблице ниже показаны значения переменной *Out* (результат обработки).

Значение *In*	Значение *Out*
In < MN	MN
MN ≤ In ≤ MX	In

Значение *In*	Значение *Out*
MX < In	MX

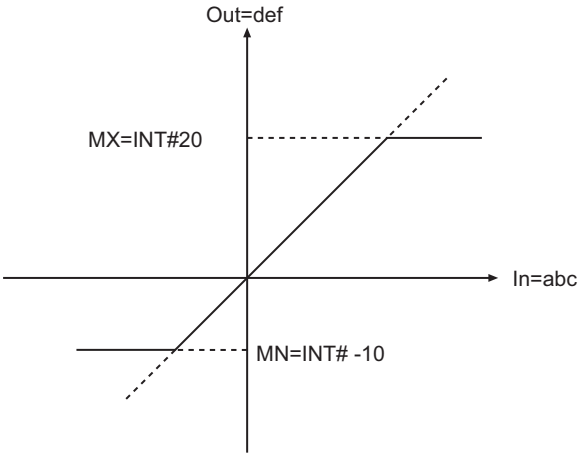
Ниже показан пример, в котором переменные *MN* и *MX* имеют значения INT#-10 и INT#20 соответственно.

LD



ST

```
def:=LIMIT(INT#-10, abc, INT#20);
```

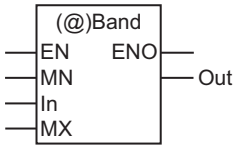


Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных переменных *In*, *MN*, *MX* и *Out* могут различаться, однако необходимо соблюдать следующие меры предосторожности.
  a) Диапазон допустимых значений *Out* должен включать в себя диапазоны допустимых значений *In*, *MN* и *MX*.
  b) Не используйте для *In*, *MN* и *MX* одновременно целые числа со знаком (SINT, INT, DINT и LINT) и целые числа без знака (USINT, UINT, UDINT и ULINT).
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *MX* меньше значения *MN*.

Band

Команда Band реализует зону нечувствительности.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Band	Зона нечувствительности	FUN		Out:=Band(MN, In, MX);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
MN	Минимальное значение	Вход	Минимальное значение зоны нечувствительности	Зависит от типа данных.	---	*1
In	Данные для управления		Данные для управления			
MX	Максимальное значение		Максимальное значение зоны нечувствительности			
Out	Результат обработки	Выход	Результат обработки	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MN										OK	OK	OK	OK	OK	OK					
In										OK	OK	OK	OK	OK	OK					
MX										OK	OK	OK	OK	OK	OK					
Out										OK	OK	OK	OK	OK	OK					

Функция

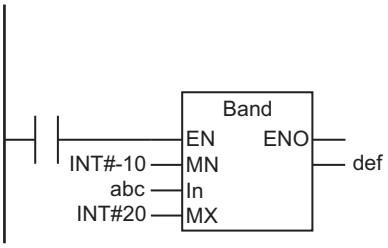
Команда Band ограничивает значение *In* (данные для управления), создавая зону нечувствительности между значением *MX* (максимальное значение) и значением *MN* (минимальное значение).

В таблице ниже показаны значения переменной *Out* (результат обработки).

Значение *In*	Значение *Out*
In < MN	*In* - *MN*
MN ≤ In ≤ MX	0
MX < In	*In* - *MX*

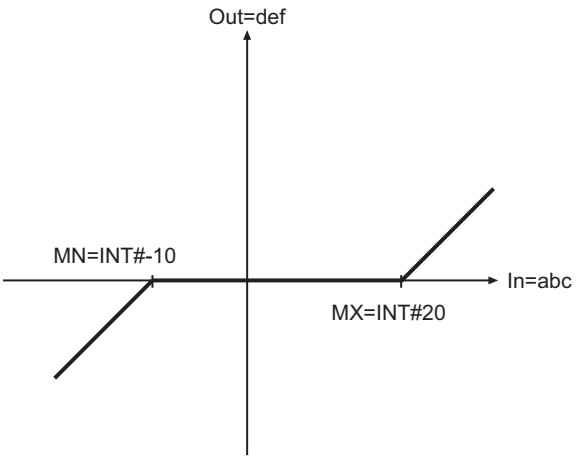
Ниже показан пример, в котором переменные *MN* и *MX* имеют значения INT#-10 и INT#20 соответственно.

LD



ST

```
def:=Band(INT#-10, abc, INT#20);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

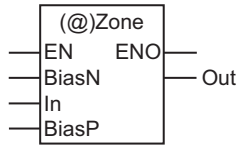
- Типы данных переменных *In*, *MN*, *MX* и *Out* могут различаться, однако необходимо соблюдать следующие меры предосторожности.
  а) Диапазон допустимых значений *Out* должен включать в себя диапазоны допустимых значений *In*, *MN* и *MX*.
- Если значение *In* является нечисловым, то и значение *Out* также является нечисловым.
- В таблице ниже приведены значения *Out* для случаев, когда *In*, *MN* и *MX* являются положительной или отрицательной бесконечностью.

Значение *In*	Значение *MN*	Значение *MX*	Значение *Out*
+ ∞	+ ∞	+ ∞	0
		-∞	Ошибка
	-∞	+ ∞	0
		-∞	+ ∞
-∞	+ ∞	+ ∞	-∞
		-∞	Ошибка
	-∞	+ ∞	0
		-∞	0

- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *MX* меньше значения *MN*.
  b) Переменная *MX* или *MN* содержит нечисловое значение.
  c) Результат выходит за допустимый диапазон *Out*.

Zone

Команда Zone добавляет значение смещения к входному значению.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Zone	Мертвая зона	FUN		Out:=Zone(BiasN, In, BiasP);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
BiasN	Отрицательное смещение	Вход	Отрицательное смещение	Зависит от типа данных.	---	*1
In	Данные для управления		Данные для управления			
BiasP	Положительное смещение		Положительное смещение			
Out	Результат обработки	Выход	Результат обработки	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
BiasN										OK	OK	OK	OK	OK	OK					
In										OK	OK	OK	OK	OK	OK					
BiasP										OK	OK	OK	OK	OK	OK					
Out										OK	OK	OK	OK	OK	OK					

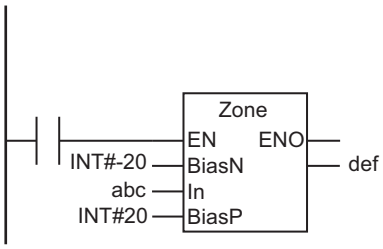
Функция

Команда Zone регулирует значение *In* (данные для управления), добавляя к нему положительное (*BiasP*) или отрицательное (*BiasN*) смещение.
В таблице ниже показаны значения переменной *Out* (результат обработки).

Значение *In*	Значение *Out*
In < 0	*In + BiasN*
In = 0	0
0 < In	*In + BiasP*

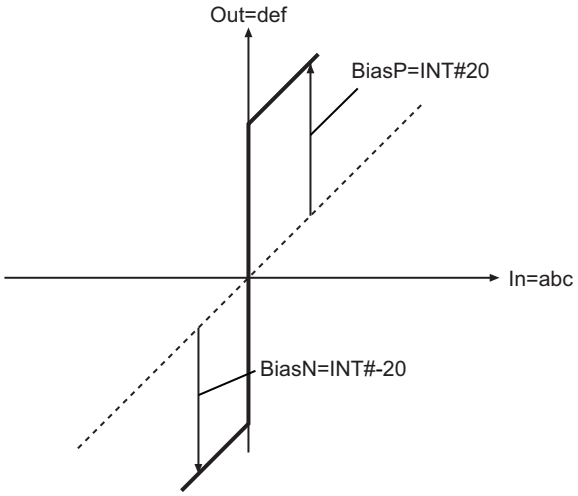
Ниже показан пример, в котором переменные *BiasP* и *BiasN* имеют значения INT#20 и INT#-20 соответственно.

LD



ST

```
def:=Zone(INT#-20, abc, INT#20);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных переменных *In*, *BiasP*, *BiasN* и *Out* могут различаться, однако необходимо соблюдать следующие меры предосторожности.
  а) Диапазон допустимых значений *Out* должен включать в себя диапазоны допустимых значений *In*, *BiasP* и *BiasN*.
- Если значение *In* является нечисловым, то и значение *Out* также является нечисловым.
- В таблице ниже приведены значения *Out* для случаев, когда *In*, *BiasP* и *BiasN* являются положительной или отрицательной бесконечностью.

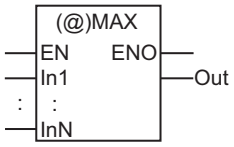
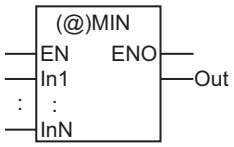
Значение *In*	Значение *BiasP*	Значение *BiasN*	Значение *Out*
+ ∞	+ ∞	+ ∞	+ ∞
		-∞	+ ∞
	-∞	+ ∞	Ошибка
		-∞	0
-∞	+ ∞	+ ∞	0
		-∞	-∞
	-∞	+ ∞	Ошибка
		-∞	-∞

- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  а) Значение *BiasP* меньше значения *BiasN*.

b) Переменная *BiasP* или *BiasN* содержит нечисловое значение.
c) Результат выходит за допустимый диапазон *Out*.

MAX и MIN

MAX : Находит наибольшее из нескольких значений (от двух до пяти).
MIN : Находит наименьшее из нескольких значений (от двух до пяти).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MAX	Максимум	FUN		Out:=MAX(In1, In2, ... , InN);
MIN	Минимум	FUN		Out:=MIN(In1, In2, ... , InN);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Данные для обработки	Вход	Данные для обработки, где N = 2...5	Зависит от типа данных.	---	0*1
Out	Результат поиска	Выход	Результат поиска	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3 и будут опущены входные параметры, подключаемые к *In1* и *In2*, то для них будут применены значения по умолчанию. Но если будет опущен входной параметр, подключаемый к *In3*, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					

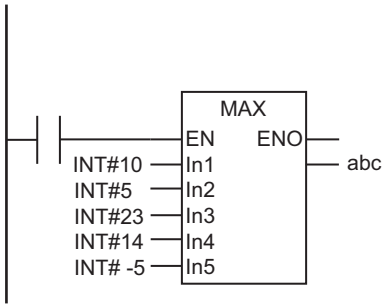
Функция

MAX

Команда MAX принимает от двух до пяти значений *In1*...*InN* (данные для обработки) и находит среди них наибольшее.

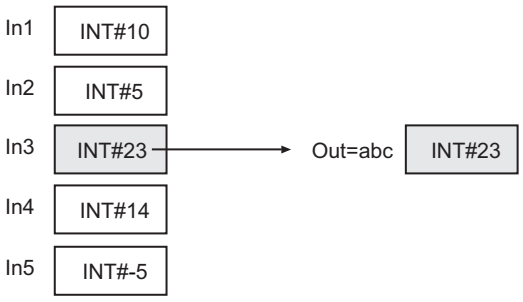
Ниже показан пример, в котором переменные *In1*, *In2*, *In3*, *In4* и *In5* имеют значения INT#10, INT#5, INT#23, INT#14 и INT#-5 соответственно.

LD



ST

```
abc:=MAX(INT#10, INT#5, INT#23, INT#14, INT#-5);
```



MIN

Команда MIN принимает от двух до пяти значений *In1*...*InN* (данные для обработки) и находит среди них наименьшее.

Дополнительная информация

Для поиска наибольшего или наименьшего из шести или более значений используйте команду *AryMax и AryMin* на стр. 2-399.

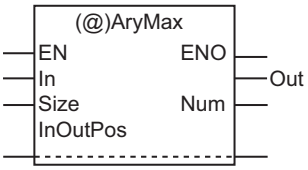
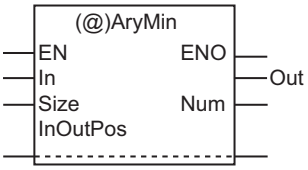
Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных переменных *In1*...*InN* и *Out* могут различаться, однако необходимо соблюдать следующие меры предосторожности.
  a) Диапазон допустимых значений *Out* должен включать в себя диапазоны допустимых значений *In1*...*InN*.
  b) Не используйте для *In1*...*InN* одновременно целые числа со знаком (SINT, INT, DINT и LINT) и целые числа без знака (USINT, UINT, UDINT и ULINT).

- Если значения *In1...InN* являются вещественными числами, желаемые результаты могут быть не получены из-за ошибки.

AryMax и AryMin

AryMax : Находит элементы с наибольшим значением в одномерном массиве.
AryMin : Находит элементы с наименьшим значением в одномерном массиве.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryMax	Максимум массива	FUN		Out:=AryMax(In, Size, InOutPos, Num);
AryMin	Минимум массива	FUN		Out:=AryMin(In, Size, InOutPos, Num);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для поиска	Вход	Массив для поиска	Зависит от типа данных.	---	*1
Size	Количество элементов для поиска		Количество элементов в In [] для поиска			1
InOutPos	Номер найденного элемента	Вход-выход	Номер элемента массива, в котором найдено значение	Зависит от типа данных.	---	---
Out	Результат поиска	Выход	Результат поиска	Зависит от типа данных.	---	---
Num	Количество найденных элементов		Количество найденных элементов			

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК *1	ОК *1	ОК *1	ОК *1	ОК *1
Size							ОК													
InOutPos							ОК													

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК *1	ОК *1	ОК *1	ОК *1	ОК *1
Num							ОК													

*1. При использовании модуля ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше можно указать следующий тип: TIME, DATE, TOD, DT и STRING.

Функция

Эти команды производят поиск среди *Size* элементов массива In[] (массив для поиска), начиная с элемента In[0].

Найденное значение присваивается переменной *Out* (результат поиска); номер элемента, в котором найдено значение, присваивается переменной *InOutPos*; количество обнаруженных элементов с таким значением присваивается переменной *Num*.
Если *Num* больше 1, то переменной *InOutPos* присваивается наименьший из номеров элементов, в которых найдено значение результата поиска.

Ниже показано, как обрабатываются значения, не являющиеся целыми и вещественными числами.

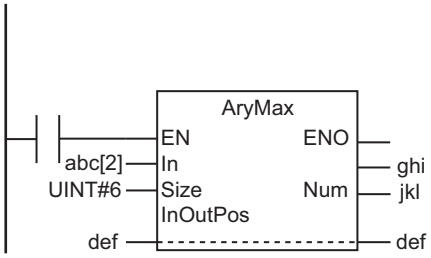
Тип данных	Описание
TIME	За большее принимается значение, которое больше в числовом выражении.
DATE, TOD или DT	За большее принимается более поздняя дата или время суток.
STRING	Применяются те же правила, что и для команд *LTascii, LEascii, GTascii и GEascii* на стр. 2-122. Подробную информацию см. на указанной странице.

AryMax

Команда AryMax находит наибольшее значение.

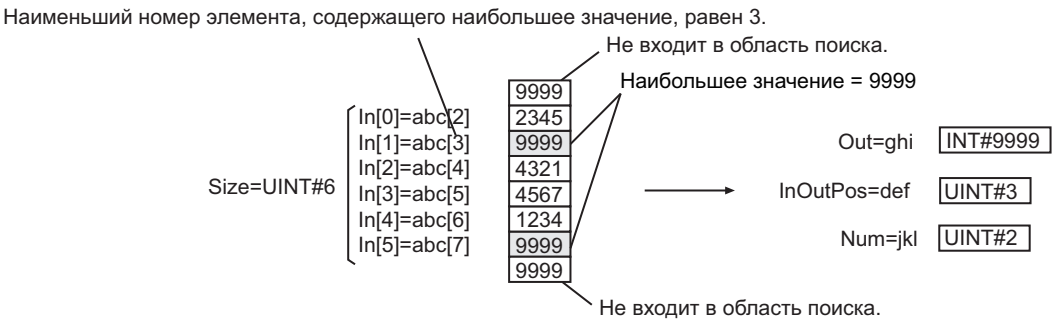
Ниже показан пример использования команды AryMax для случая, когда *Size* = UINT#6.
В качестве входного параметра на вход In[] подается abc[2], так что поиск начинается с abc[2].

LD



ST

```
ghi:=AryMax(abc[2], UINT#6, def, jkl);
```



AryMin

Команда AryMin находит наименьшее значение.

Дополнительная информация

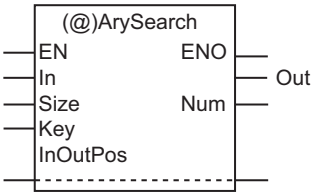
При сравнении значений типа TIME, DT или TOD необходимо предварительно отрегулировать их точность, чтобы сравниваемые значения этих типов данных имели одинаковую точность. Для определения точности значений можно использовать следующие команды: *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 и *TruncTod* на стр. 2-757.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если типы данных In[] и *Out* отличаются, проследите, чтобы диапазон допустимых значений *Out* вмещал в себя диапазон допустимых значений In[].
- Если In[] содержит вещественное число, желаемый результат может быть не получен из-за ошибки.
- Массив In[] должен быть одномерным.
- Если значение *Size* равно 0, то значения *Out* и *Num* также равны 0. Значение *InOutPos* при этом не изменяется.
- Если In[] содержит строковые данные (STRING), а значение *Size* равно 0, *Out* содержит только нулевые символы (NULL).
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Size* находится за пределами допустимого диапазона.
  b) Значение *Size* приводит к выходу за область массива In[].
  c) In[] не является одномерным массивом.
  d) In[] содержит строковые данные (STRING) и не завершается символом NULL.

ArySearch

Команда ArySearch выполняет поиск указанного значения в одномерном массиве.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ArySearch	Поиск по массиву	FUN		Out:=ArySearch(In, Size, Key, InOutPos, Num);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для поиска	Вход	Массив для поиска	Зависит от типа данных.	---	*1
Size	Количество элементов для поиска		Количество элементов в In [] для поиска			1
Key	Ключ поиска		Значение для поиска	Зависит от типа данных.		*1
InOutPos	Номер найденного элемента	Вход-выход	Номер элемента массива, в котором найдено значение	Зависит от типа данных.	---	---
Out	Результат поиска	Выход	Результат поиска	Зависит от типа данных.	---	---
Num	Количество найденных элементов		Количество найденных элементов			

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK *1	OK *1	OK *1	OK *1	OK
	Также допускается указывать массивы перечислений.																			
Size							OK													
Key	Тип данных должен быть таким же, как у элементов массива In[].																			
InOutPos							OK													
Out	OK																			
Num							OK													

*1. При использовании модуля ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше можно указать значение следующего типа: TIME, DATE, TOD и DT.

Функция

Команда ArySearch производит поиск элементов, содержащих значение *Key* (ключ поиска), среди *Size* элементов одномерного массива. Поиск начинается с элемента In[0].

В следующей таблице приведены значения переменных *Out* (результат поиска), *InOutPos* (номер найденного элемента) и *Num* (количество найденных элементов).

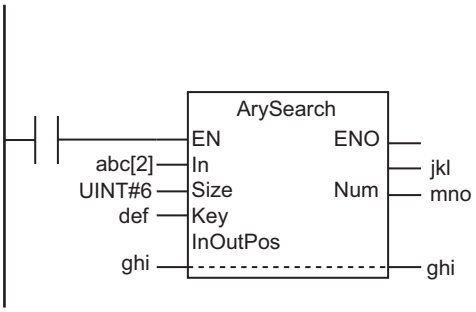
Элемент с тем же значением, что и *Key*	Out	InOutPos	Num
Существует.	ИСТИНА	Наименьший из номеров элементов, содержащих значение *Key*	Количество элементов, содержащих значение *Key*
Не существует.	ЛОЖЬ	Не меняется.	0

Ниже показано, как обрабатываются значения, не являющиеся целыми и вещественными числами.

Тип данных	Описание
TIME	За большее принимается значение, которое больше в числовом выражении.
DATE, TOD или DT	За большее принимается более поздняя дата или время суток.

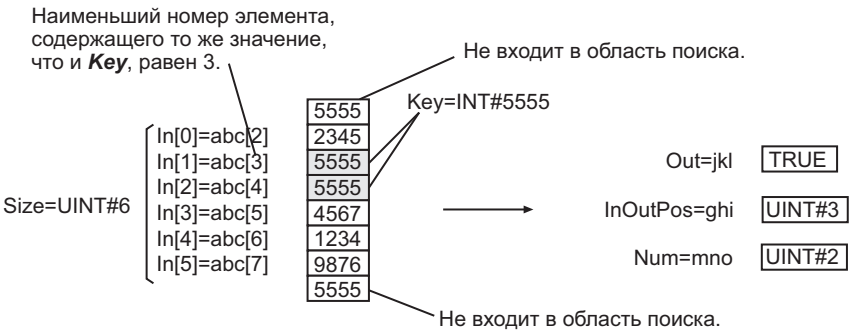
В показанном ниже примере программы *Size* = UINT#6.
В качестве входного параметра на вход In[] подается abc[2], так что поиск начинается с abc[2].

LD



ST

```
jkl:=ArySearch(abc[2], UINT#6, def, ghi, mno);
```



Дополнительная информация

При сравнении значений типа TIME, DT или TOD необходимо предварительно отрегулировать их точность, чтобы сравниваемые значения этих типов данных имели одинаковую точность. Для определения точности значений можно использовать следующие команды: *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 и *TruncTod* на стр. 2-757.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Массив In[] должен быть одномерным.
- Проследите, чтобы тип данных параметра *Key* совпадал с типом данных элементов массива In[].
- Если значение *Size* равно 0, то значения *Out* и *Num* также равны 0. Значение *InOutPos* при этом не изменяется.
- Всегда используйте переменную в качестве входного параметра, передаваемого в *Key*. При передаче константы произойдет ошибка сборки.
- Если параметр *Key* является перечислением, в него невозможно напрямую передать перечислитель. В случае непосредственной передачи перечислителя произойдет ошибка сборки.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значения на выходах *Out*, *Num* и *InOutPos* не изменятся.
  a) Значение *Size* приводит к выходу за область массива In[].
  b) In[] содержит строковые данные (STRING) и не завершается символом NULL.
  c) In[] не является одномерным массивом.

Команды перемещения данных

Команда	Имя	Стр.
MOVE	Перемещение	стр. 2-406
MoveBit	Перемещение бита	стр. 2-409
MoveDigit	Перемещение разряда	стр. 2-411
TransBits	Перемещение битов	стр. 2-413
MemCopy	Копирование памяти	стр. 2-416
SetBlock	Заполнение блока	стр. 2-418
Exchange	Обмен данными	стр. 2-420
AryExchange	Обмен данными массивов	стр. 2-422
AryMove	Перемещение массива	стр. 2-424
Clear	Инициализация	стр. 2-426
Copy**ToNum (Битовая строка в целое число со знаком)	Группа копирования комбинации битов (Битовая строка в целое число со знаком)	стр. 2-429
Copy**To*** (Битовая строка в вещественное число)	Группа копирования комбинации битов (Битовая строка в вещественное число)	стр. 2-431
CopyNumTo** (Целое число со знаком в битовую строку)	Группа копирования комбинации битов (Целое число со знаком в битовую строку)	стр. 2-433
CopyNumTo** (Целое число со знаком в вещественное число)	Группа копирования комбинации битов (Целое число со знаком в вещественное число)	стр. 2-435
Copy**To*** (Вещественное число в битовую строку)	Группа копирования комбинации битов (Вещественное число в битовую строку)	стр. 2-437
Copy**ToNum (Вещественное число в целое число со знаком)	Группа копирования комбинации битов (Вещественное число в целое число со знаком)	стр. 2-439

MOVE

Команда MOVE перемещает значение константы или переменной в другую переменную.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MOVE	Перемещение	FUN	(@)MOVE EN ENO In Out	Out:=In;

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник перемещения	Вход	Источник перемещения	Зависит от типа данных.	---	*1
Out	Адресат перемещения	Выход	Адресат перемещения	Зависит от типа данных.	---	*1

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

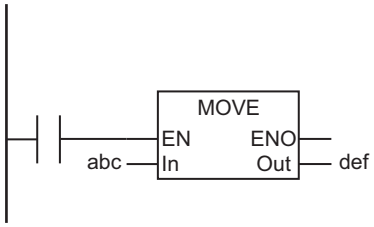
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать перечисление, массив, элемент массива, структуру или член структуры.																			
Out	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Если переменная *In* является перечислением, элементом массива, структурой или членом структуры, тип данных должен быть таким же, как у *In*. Если переменная *In* является массивом, должно быть массивом с тем же типом данных, того же размера и с теми же индексами, что и у *In*.																			

Функция

Команда MOVE перемещает значение переменной *In* (источник перемещения) в переменную *Out* (адресат перемещения).
Входной параметр, который передается в *In*, может быть переменной или константой. Для *In* можно указать перечисление, массив, элемент массива, структуру или член структуры.

Пример программы представлен на рисунке ниже. Содержимое переменной *abc* перемещается в переменную *def*.

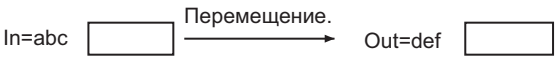
LD



ST

```
def:=abc;
```

Команда MOVE перемещает значение ***In*** в ***Out***.

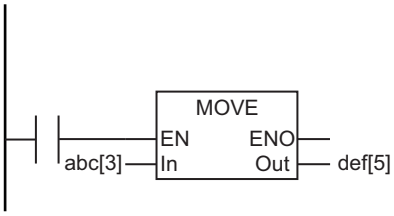


Дополнительная информация

- При перемещении массива можно переместить либо один элемент, либо все элементы массива.
  Чтобы переместить только один элемент, добавьте индекс к имени переменной массива.
  Чтобы переместить весь массив, не добавляйте индекс к имени переменной массива.

Перемещение одного элемента массива

LD

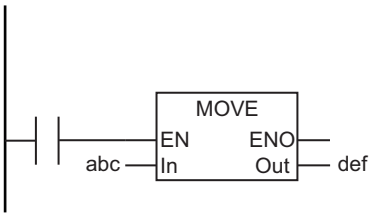


ST

```
def[5]:=abc[3];
```

Перемещение всех элементов массива

LD



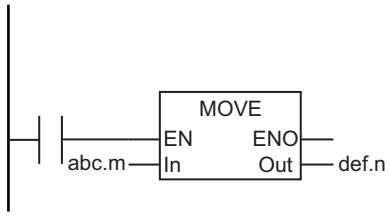
ST

```
def:=abc;
```

- При перемещении структуры можно переместить либо один член, либо все члены структуры.
  Чтобы переместить только один член структуры, укажите его.
  Чтобы переместить всю структуру, укажите только имя структуры.

Перемещение одного члена структуры

LD

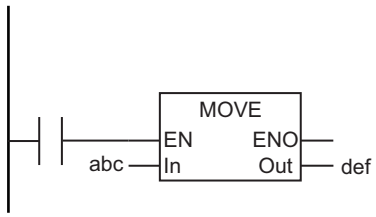


ST

```
def.n:=abc.m;
```

Перемещение всей структуры

LD



ST

```
def:=abc;
```

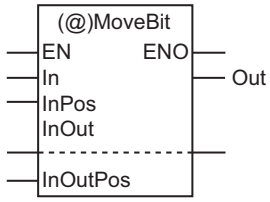
- Для перемещения всего массива можно использовать команду MemCopy, которая делает это быстрее, чем команда MOVE.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных переменных *In* и *Out* могут различаться, однако они должны принадлежать к одной группе типов данных (см. ниже). Диапазон допустимых значений *Out* должен включать в себя диапазон допустимых значений *In*.
  a) BOOL, WORD, DWORD и LWORD
  b) USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL и LREAL
- Если переменная *In* является перечислением, элементом массива, структурой или членом структуры, переменная *Out* должна иметь тот же тип данных, что и *In*.
- Если переменная *In* является массивом, для переменной *Out* должен использоваться массив того же типа данных, того же размера и с теми же индексами, что и у массива *In*.

MoveBit

Команда MoveBit перемещает один бит в битовой строке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MoveBit	Перемещение бита	FUN		MoveBit(In, InPos, InOut, InOutPos);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник перемещения	Вход	Источник перемещения	Зависит от типа данных.	---	*1
InPos	Исходный бит для перемещения		Позиция исходного бита в *In*	От 0 до количества битов в *In* - 1		0
InOutPos	Целевой бит для перемещения		Позиция целевого бита в *InOut*	От 0 до количества битов в *InOut* - 1		
InOut	Адресат перемещения	Вход-выход	Адресат перемещения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

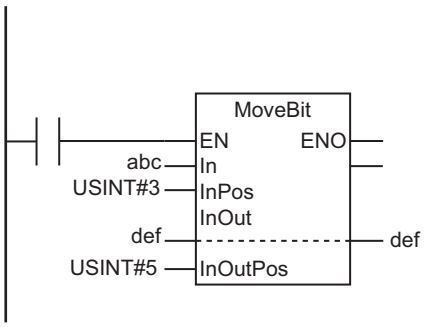
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
InPos						OK														
InOutPos						OK														
InOut		OK	OK	OK	OK															
Out	OK																			

Функция

Команда MoveBit перемещает один бит переменной *In* (источник перемещения), находящийся в позиции *InPos* (позиция исходного бита), в бит переменной *InOut* (адресат перемещения), находящийся в позиции *InOutPos* (позиция целевого бита).

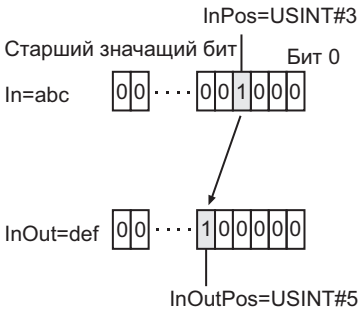
Ниже показан пример для случая, когда *InPos* = USINT#3, а *InOutPos* = USINT#5.

LD



ST

```
MoveBit(abc, USINT#3, def, USINT#5);
```

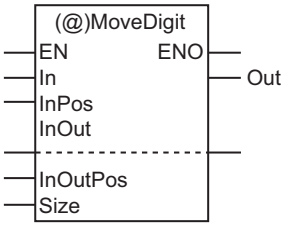


Меры предосторожности для обеспечения надлежащей эксплуатации

- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *InOut* не изменится.
  a) Значение *InPos* находится за пределами допустимого диапазона.
  b) Значение *InOutPos* находится за пределами допустимого диапазона.

MoveDigit

Команда MoveDigit перемещает разряды (по 4 бита на разряд) в битовой строке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MoveDigit	Перемещение разряда	FUN		MoveDigit(In, InPos, InOut, InOutPos, Size);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник перемещения	Вход	Источник перемещения	Зависит от типа данных.	---	*1
InPos	Исходный разряд для перемещения		Позиция исходного разряда в *In*	От 0 до количества битов в *In*/4 - 1		0
InOutPos	Целевой разряд для перемещения		Позиция целевого разряда в *InOut*	От 0 до количества битов в *InOut*/4 - 1		
Size	Количество разрядов		Количество перемещаемых разрядов	От 0 до количества битов в *In*/4		1
InOut	Адресат перемещения	Вход-выход	Адресат перемещения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

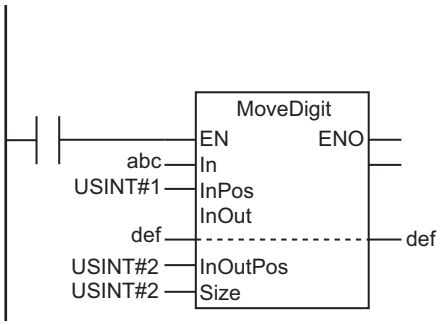
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
InPos						OK														
InOutPos						OK														
Size						OK														
InOut		OK	OK	OK	OK															
Out	OK																			

Функция

Команда MoveDigit перемещает *Size* разрядов переменной *In* (источник перемещения), начиная с разряда в позиции *InPos* (позиция исходного разряда), в соответствующие разряды переменной *InOut* (адресат перемещения), начиная с разряда в позиции *InOutPos* (позиция целевого разряда). Один разряд представляет четыре бита.

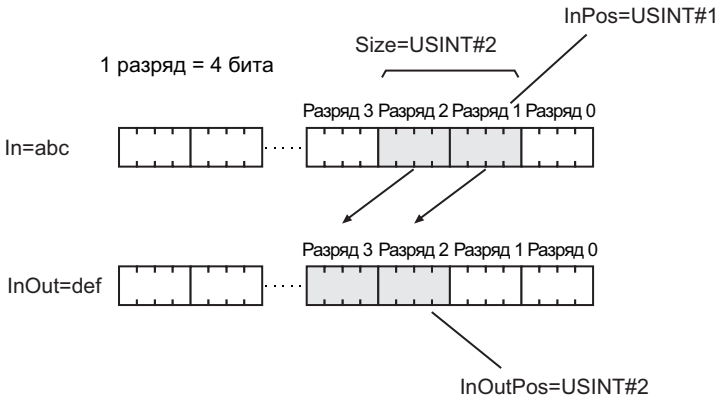
Ниже показан пример для случая, когда *InPos* = USINT#1, *InOutPos* = USINT#2, а *Size* = USINT#2.

LD



ST

```
MoveDigit(abc, USINT#1, def, USINT#2, USINT#2);
```

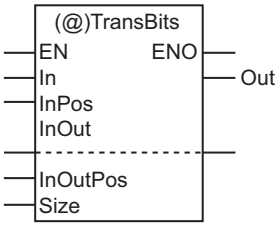


Меры предосторожности для обеспечения надлежащей эксплуатации

- Если заданная позиция целевого разряда приводит к выходу за самый старший разряд переменной *InOut*, остальные разряды записываются в младшие разряды *InOut*.
- Если заданная позиция исходного разряда приводит к выходу за самый старший разряд переменной *In*, остальные разряды перемещаются в младшие разряды *In*.
- Если *Size* = 0, выход *Out* будет содержать ИСТИНА, а значение *InOut* не изменится.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *InOut* не изменится.
  a) Значение *InPos* находится за пределами допустимого диапазона.
  b) Значение *InOutPos* находится за пределами допустимого диапазона.
  c) Значение *Size* находится за пределами допустимого диапазона.

TransBits

Команда TransBits перемещает один или несколько битов в битовой строке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TransBits	Перемещение битов	FUN		TransBits(In, InPos, InOut, InOutPos, Size);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник перемещения	Вход	Источник перемещения	Зависит от типа данных.	---	*1
InPos	Исходный бит для перемещения		Позиция исходного бита в *In*	От 0 до количества битов в *In* - 1		0
InOutPos	Целевой бит для перемещения		Позиция целевого бита в *InOut*	От 0 до количества битов в *InOut* - 1		
Size	Количество битов		Количество перемещаемых битов	От 0 до количества битов в *In*		1
InOut	Адресат перемещения	Вход-выход	Адресат перемещения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

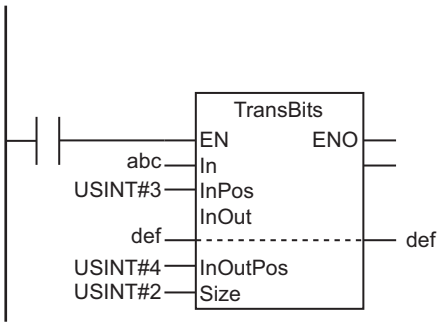
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
InPos						OK														
InOutPos						OK														
Size						OK														
InOut		OK	OK	OK	OK															
Out	OK																			

Функция

Команда TransBits перемещает *Size* битов переменной *In* (источник перемещения), начиная с позиции *InPos* (позиция исходного бита), в соответствующие биты переменной *InOut* (адресат перемещения), начиная с позиции *InOutPos* (позиция целевого бита).

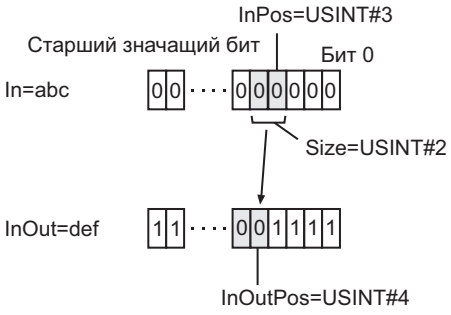
Ниже показан пример для случая, когда *InPos* = USINT#3, *InOutPos* = USINT#4, а *Size* = USINT#2.

LD



ST

```
TransBits(abc, USINT#3, def, USINT#4, USINT#2);
```



Дополнительная информация

Биты в источнике перемещения могут перекрываться с битами в адресате перемещения.

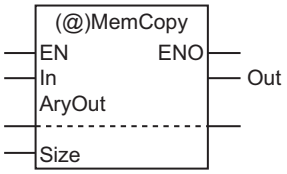
Меры предосторожности для обеспечения надлежащей эксплуатации

- Задавайте параметры команды так, чтобы позиции исходного и целевого битов не выходили за позицию самого старшего бита в переменных *In* или *InOut* соответственно. В противном случае произойдет ошибка и команда работать не будет.
- Если значение *Size* равно 0, ничего не перемещается.
- Содержимое остальных битов *InOut*, которые не участвуют в операции перемещения, не изменяется.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *InOut* не изменится.
  a) Значение *InPos* находится за пределами допустимого диапазона.
  b) Значение *InOutPos* находится за пределами допустимого диапазона.
  c) Значение *Size* находится за пределами допустимого диапазона.
  d) Значение *InPos* или *Size* больше, чем количество битов в *In*.

e) Значение *InOutPos* или *Size* больше, чем количество битов в *InOut*.

MemCopy

Команда MemCopy перемещает один или несколько элементов массива. Источник перемещения и адресат перемещения должны иметь одинаковый тип данных.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MemCopy	Копирование памяти	FUN		MemCopy(In, AryOut, Size);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Исходный массив для перемещения	Вход	Исходный массив для перемещения	Зависит от типа данных.	---	*1
Size	Количество элементов		Количество перемещаемых элементов массива			1
AryOut[] (массив)	Целевой массив для перемещения	Вход-выход	Целевой массив для перемещения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

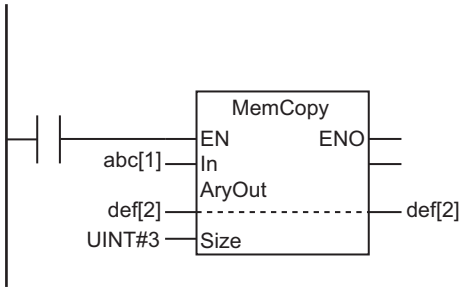
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также допускается указывать массивы перечислений или структур.																			
Size							ОК													
AryOut[] (массив)	Должен быть массивом с тем же типом данных, что и In[].																			
Out	ОК																			

Функция

Команда MemCopy перемещает *Size* элементов массива In[] (исходный массив для перемещения), начиная с элемента In[0], в массив AryOut[] (целевой массив для перемещения), начиная с элемента AryOut[0].

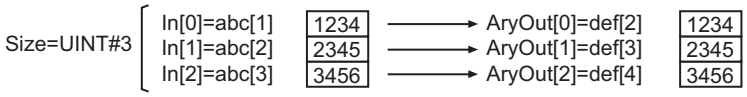
В показанном ниже примере программы *Size* = UINT#3.

LD



ST

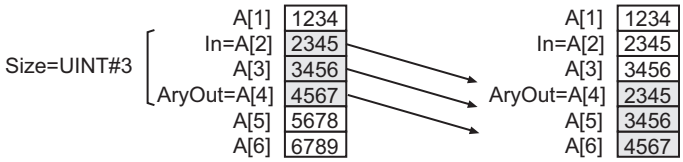
```
MemCopy(abc[1], def[2], UINT#3);
```



Дополнительная информация

- В параметры In[] и AryOut[] можно ввести один и тот же массив и указать разные позиции. Исходные и целевые данные могут перекрываться.

Ниже приведен пример для случая, когда *In* = A[2], *AryOut* = A[4], а *Size* = UINT#3.



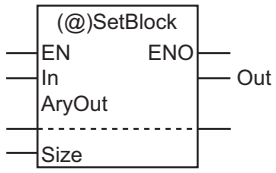
- Если исходный и целевой массивы относятся к разным типам данных, используйте команду *AryMove* на стр. 2-424.
- Если же исходный и целевой массивы имеют одинаковый тип данных, используйте данную команду, так как она работает быстрее команды AryMove.
- Для перемещения переменных, не являющихся массивами, используйте команду *MOVE* на стр. 2-406.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте одинаковый тип данных для массивов In[] и AryOut[]. При использовании разных типов данных произойдет ошибка сборки.
- Если In[] и AryOut[] являются массивами строковых значений (STRING), их размеры должны быть одинаковыми.
- Если *Size* = 0, состояние выхода *Out* поменяется на ИСТИНА, а содержимое AryOut[] не изменится.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) *Size* приводит к выходу за область массива In[].
  b) *Size* приводит к выходу за область массива AryOut[].

SetBlock

Команда SetBlock перемещает значение переменной или константы в один или несколько элементов массива.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SetBlock	Заполнение блока	FUN		SetBlock(In, AryOut, Size);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник перемещения	Вход	Источник перемещения	Зависит от типа данных.	---	*1
Size	Количество элементов		Количество перемещаемых элементов массива			1
AryOut[] (массив)	Целевой массив для перемещения	Вход-выход	Целевой массив для перемещения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

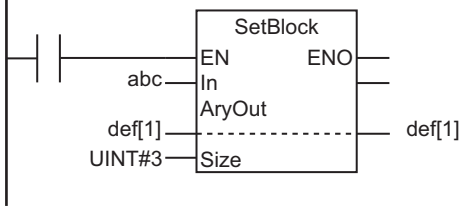
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также можно указать перечисление, структуру или член структуры.																			
Size							ОК													
AryOut[] (массив)	Должен представлять собой массив с элементами того же типа данных, что и значение *In*.																			
Out	ОК																			

Функция

Команда SetBlock перемещает значение *In* (источник перемещения) в *Size* элементов массива AryOut[] (целевой массив для перемещения), начиная с элемента AryOut[0].

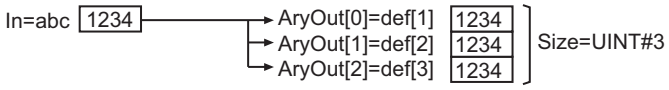
В показанном ниже примере программы *Size* = UINT#3.

LD



ST

```
SetBlock(abc, def[1], UINT#3);
```

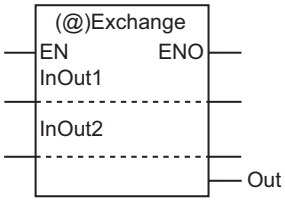


Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте одинаковый тип данных для переменных *In* и AryOut[]. При использовании разных типов данных произойдет ошибка сборки.
- Если *In* и AryOut[] содержат строковые данные (STRING), их размеры должны быть одинаковыми.
- Если *Size* = 0, выход *Out* будет содержать ИСТИНА, а содержимое AryOut[] не изменится.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение *Size* приводит к выходу за область массива AryOut[].

Exchange

Команда Exchange меняет местами значения двух переменных.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Exchange	Обмен данными	FUN		Exchange(InOut1, InOut2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
InOut1 и InOut2	Данные для обмена	Вход-выход	Данные для обмена	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

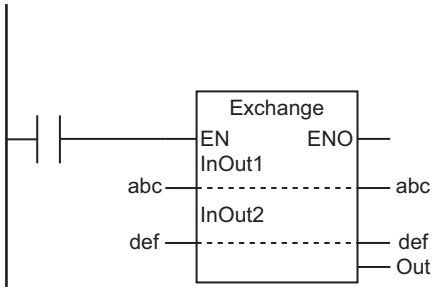
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut1	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также можно указать перечисление, структуру или член структуры.																			
InOut2	Тип данных должен быть таким же, как у *InOut1*.																			
Out	ОК																			

Функция

Команда Exchange меняет местами значения переменных *InOut1* и *InOut2* (данные для обмена). Для переменных *InOut1* и *InOut2* можно указывать перечисления, структуры или члены структуры.

Пример программы представлен на рисунке ниже. Переменные *abc* и *def* меняются своими значениями.

LD



ST

```
Exchange(abc, def);
```

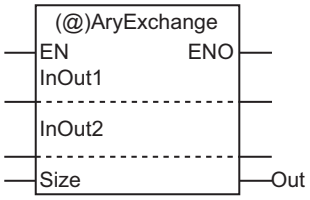
Команда Exchange меняет местами значения переменных ***InOut1*** и ***InOut2***.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных *InOut1* и *InOut2* должны быть одинаковыми. При использовании разных типов данных произойдет ошибка сборки.
- Если области, указанные переменными *InOut1* и *InOut2*, перекрывают друг друга, результат выполнения команды будет неопределенным.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае происходит ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значения на выходах *InOut1* и *InOut2* не изменятся.
  a) Переменные *InOut1* и *InOut2* содержат строковые значения (STRING) и одна из них не может вместить в себя значение другой из-за разницы в размерах.

AryExchange

Команда AryExchange меняет местами элементы двух массивов.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryExchange	Обмен данными массивов	FUN		AryExchange(InOut1, InOut2, Size);

Переменные

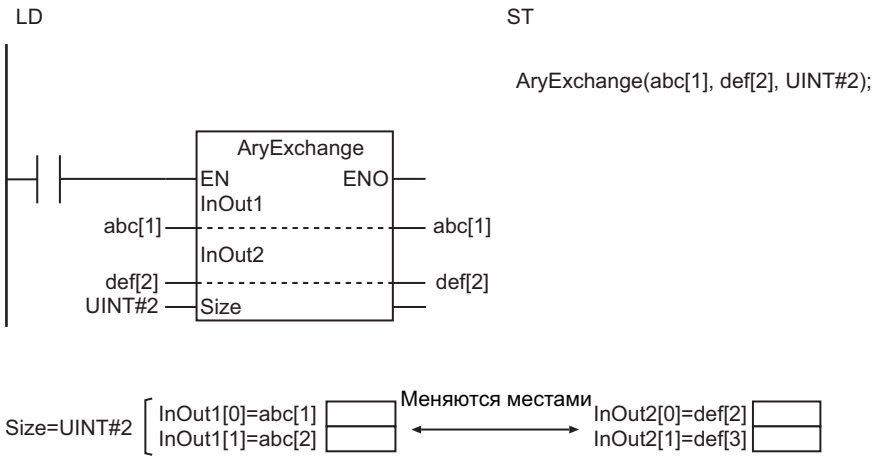
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Size	Количество элементов	Вход	Количество элементов для обмена	Зависит от типа данных.	---	1
InOut1[] и InOut2[] (массивы)	Массивы для обмена	Вход-выход	Массивы для обмена	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size							ОК													
InOut1[] (массив)	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также допускается указывать массивы перечислений или структур.																			
InOut2[] (массив)	Должен быть массивом с тем же типом данных, что и InOut1[].																			
Out	ОК																			

Функция

Команда AryExchange меняет местами *Size* элементов массива InOut1[] (массив для обмена), начиная с элемента InOut1[0], и *Size* элементов массива InOut2[], начиная с элемента InOut2[0].

В показанном ниже примере программы *Size* = UINT#2.



Дополнительная информация

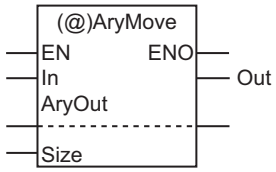
- Для присвоения значений констант переменным используйте команду *MOVE* на стр. 2-406.
- Для копирования значений переменных в другие переменные используйте команду *MemCopy* на стр. 2-416.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте одинаковый тип данных для массивов InOut1[] и InOut2[]. При использовании разных типов данных произойдет ошибка сборки.
- Если *Size* = *0*, выход *Out* будет содержать ИСТИНА, а содержимое массивов InOut1[] и InOut2[] не изменится.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое массивов InOut1[] и InOut2[] не изменится.
  a) Значение *Size* превышает количество элементов массива InOut1[] или InOut2[].
  b) Массивы InOut1[] и InOut2[] являются массивами строкового типа (STRING), и длина строки элемента в одном массиве превышает длину строки соответствующего элемента в другом массиве.
  c) Массивы InOut1[] и InOut2[] являются массивами строкового типа (STRING), и какой-либо элемент не заканчивается символом NULL.

AryMove

Команда AryMove перемещает один или несколько элементов массива. Тип данных исходного массива может отличаться от типа данных целевого массива.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryMove	Перемещение массива	FUN		AryMove(In, AryOut, Size);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Исходный массив для перемещения	Вход	Массив для перемещения	Зависит от типа данных.	---	*1
Size	Количество элементов		Количество перемещаемых элементов			1
AryOut[] (массив)	Результирующий массив для перемещения	Вход-выход	Результирующий массив для перемещения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

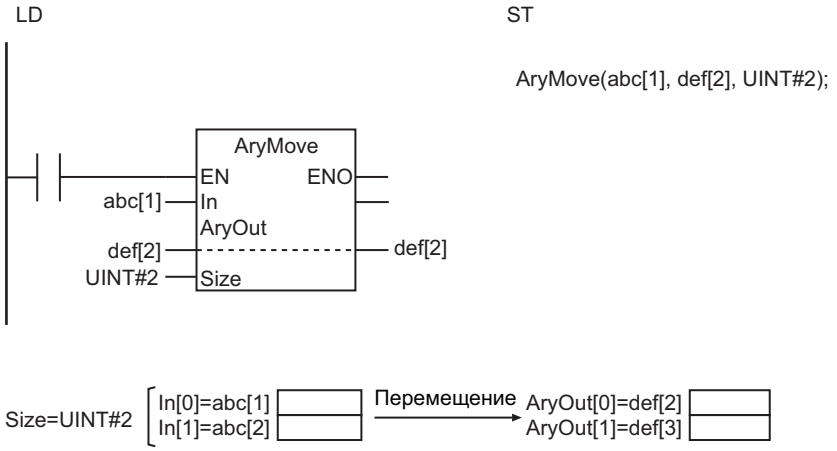
*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также допускается указывать массивы перечислений или структур.																			
Size							OK													
AryOut[] (массив)	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также допускается указывать массивы перечислений или структур.																			
Out	OK																			

Функция

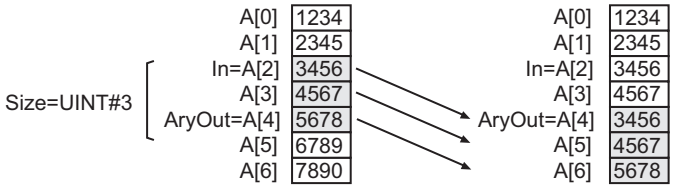
Команда AryMove перемещает *Size* элементов массива In[] (исходный массив для перемещения), начиная с элемента In[0], в массив AryOut[] (результирующий массив для перемещения), начиная с элемента AryOut[0].
Типы данных массивов In[] и AryOut[] могут различаться.

В показанном ниже примере программы *Size* = UINT#2.



Дополнительная информация

- Если типы данных массивов In[] и AryOut[] одинаковы, можно использовать команду MemCopy. Она выполняется быстрее.
- В параметры In[] и AryOut[] можно ввести один и тот же массив. Кроме того, исходные и целевые данные при перемещении могут перекрываться. Ниже приведен пример для случая, когда In[0] = A[2], AryOut[0] = A[4], а *Size* = UINT#3.

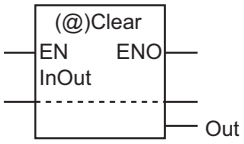


Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных массивов In[] и AryOut[] могут различаться, однако они должны принадлежать к одной группе типов данных, которые указаны ниже. Диапазон допустимых значений AryOut[] должен включать в себя диапазон допустимых значений In[].
  a) BOOL, WORD, DWORD и LWORD
  b) USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL и LREAL
- Если In[] является массивом структур, для массивов In[] и AryOut[] следует использовать одинаковый тип данных.
- Если *Size* = *0*, выход *Out* будет содержать ИСТИНА, а содержимое AryOut[] не изменится.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение *Size* превосходит размер массива In[] или AryOut[].
  b) Массивы In[] и AryOut[] являются массивами строкового типа (STRING), и длина строки какого-либо перемещаемого элемента In[] превышает размер соответствующего элемента в AryOut[].

Clear

Команда Clear инициализирует переменную.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Clear	Инициализация	FUN		Clear(InOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
InOut	Данные для инициализации	Вход-выход	Данные для инициализации	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также можно указать перечисление, массив, элемент массива, структуру или член структуры.																			
Out	ОК																			

Функция

Команда Clear инициализирует значение переменной *InOut* (данные для инициализации).
Если для переменной задан атрибут начального значения (Initial Value), то применяется начальное значение. Если атрибут начального значения не задан, применяется начальное значение по умолчанию, соответствующее типу данных переменной *InOut*.
Если *InOut* является внешней переменной, то начальное значение по умолчанию для типа данных *InOut* используется независимо от атрибута начального значения соответствующей глобальной переменной.

Ниже приведены значения по умолчанию для разных типов данных.

Тип данных	Начальное значение по умолчанию
BOOL	ЛОЖЬ
BYTE, WORD, DWORD или LWORD	16#0
USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL или LREAL	0

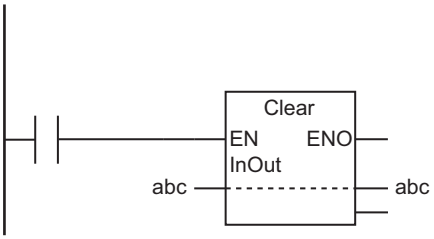
Тип данных	Начальное значение по умолчанию
TIME	T#0ms
DATE	D#1970-1-1
TOD	TOD#0:0:0
DT	DT#1970-1-1-0:0:0
STRING	''

Если *InOut* является массивом, элементом массива, структурой или членом структуры, данные обрабатываются следующим образом.

InOut	Обработка
Массив	Инициализируются все элементы массива.
Элемент массива	Инициализируется только указанный элемент.
Структура	Инициализируются все члены структуры.
Член структуры	Инициализируется только указанный член структуры.

Пример программы представлен на рисунке ниже. Инициализируется значение переменной *abc*. Например, если значение переменной *abc* равно INT#100, то после инициализации оно будет равно INT#0.

LD



ST

```
Clear(abc);
```

Команда Clear инициализирует значение переменной ***InOut***.
Тип данных ***abc*** — INT, поэтому значение ***abc*** будет равно INT#0.

Дополнительная информация

- Если *InOut* является массивом, который используется в качестве стека, то при выполнении этой команды также присвойте значение 0 переменной, которая управляет количеством элементов, хранящихся в стеке.
- Если с помощью этой команды инициализируется переменная данных кулачка, она не будет содержать данные, сохраненные с помощью команды MC_SaveCamTable. Она будет содержать только нули.

Меры предосторожности для обеспечения надлежащей эксплуатации

- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- Для инициализации переменной-перечисления используйте атрибут Initial Value (Начальное значение). Если атрибут начального значения не задан, то значение переменной-перечисления будет равно 0.

- Не выполняйте операцию, которая соответствует всем указанным ниже условиям. Эта операция непредсказуема.
  a) Один элемент массива типа BOOL передается как входная-выходная переменная в функцию или функциональный блок.
  b) В функции или функциональном блоке выполняется команда Clear.
  c) В качестве параметра, передаваемого в команду Clear, используется входная-выходная переменная, получившая элемент указанного выше массива типа BOOL.

Copy**ToNum (Битовая строка в целое число со знаком)

Команда Copy**ToNum копирует содержимое битовой строки непосредственно в целое число со знаком.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Copy**ToNum	Группа копирования комбинации битов (Битовая строка в целое число со знаком)	FUN	(@)Copy**ToNum EN ENO In Out "**" — тип данных, относящийся к битовым строкам.	Out:=Copy**ToNum(In); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник копирования	Вход	Источник копирования	Зависит от типа данных.	---	0
Out	Адресат копирования	Выход	Адресат копирования	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Out	Должен использоваться целочисленный тип данных со знаком того же размера, что и у типа данных переменной *In*.																			

Функция

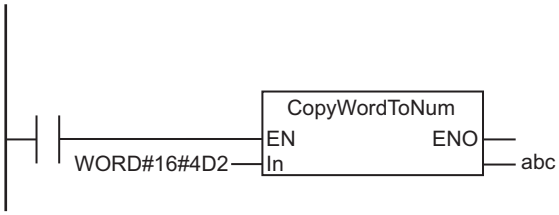
Группа команд Copy**ToNum копирует содержимое *In* (источник копирования) непосредственно в *Out* (адресат копирования).

В группу входят четыре команды для разных комбинаций типов данных *In* и *Out*, которые перечислены ниже.

In	Out	Команда
BYTE	SINT	CopyByteToNum
WORD	INT	CopyWordToNum
DWORD	DINT	CopyDwordToNum
LWORD	LINT	CopyLwordToNum

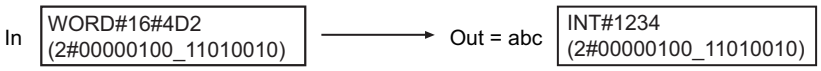
Ниже показан пример для команды CopyWordToNum, в котором *In* = WORD#16#4D2.

LD



ST

```
abc:=CopyWordToNum(WORD#16#4D2);
```



Copy**To*** (Битовая строка в вещественное число)

Команда Copy**To*** копирует содержимое битовой строки непосредственно в вещественное число.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Copy**To***	Группа копирования комбинации битов (Битовая строка в вещественное число)	FUN	(@)CopyDwordToReal EN ENO In Out (@)CopyLwordToLreal EN ENO In Out	Out:=CopyDwordToReal(In); или Out:=CopyLwordToLreal(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник копирования	Вход	Источник копирования	Зависит от типа данных.	---	0
Out	Адресат копирования	Выход	Адресат копирования	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In				OK	OK															
Out	Если *In* — значение типа DWORD, то должен использоваться тип REAL, а если LWORD, то LREAL.																			

Функция

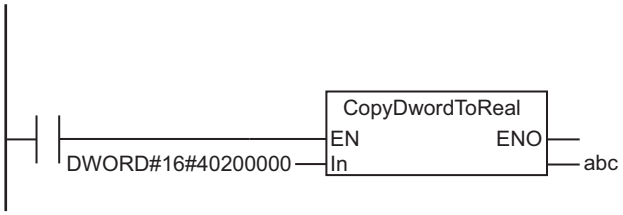
Группа команд Copy**To*** копирует содержимое *In* (источник копирования) непосредственно в *Out* (адресат копирования).

В группу входят две команды для разных комбинаций типов данных *In* и *Out*, которые перечислены ниже.

In	Out	Команда
DWORD	REAL	CopyDwordToReal
LWORD	LREAL	CopyLwordToLreal

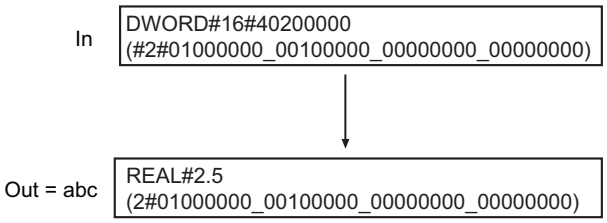
Ниже показан пример для команды CopyDwordToReal, в котором *In* = DWORD#16#40200000.

LD



ST

```
abc:=CopyDwordToReal(DWORD#16#40200000);
```



CopyNumTo** (Целое число со знаком в битовую строку)

Команда CopyNumTo** копирует содержимое целого числа со знаком непосредственно в битовую строку.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CopyNumTo**	Группа копирования комбинации битов (Целое число со знаком в битовую строку)	FUN	(@)CopyNumTo** EN ENO In Out "**" — тип данных, относящийся к битовым строкам.	Out:=CopyNumTo**(In); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник копирования	Вход	Источник копирования	Зависит от типа данных.	---	0
Out	Адресат копирования	Выход	Адресат копирования	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In										OK	OK	OK	OK							
Out	Должен использоваться тип данных «битовая строка» того же размера, что и у типа данных переменной *In*.																			

Функция

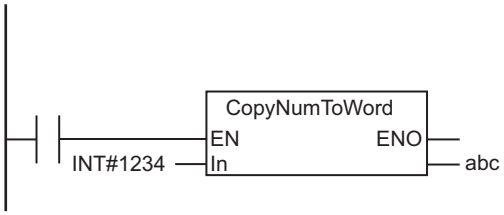
Группа команд CopyNumTo** копирует содержимое *In* (источник копирования) непосредственно в *Out* (адресат копирования).

В группу входят четыре команды для разных комбинаций типов данных *In* и *Out*, которые перечислены ниже.

In	Out	Команда
SINT	BYTE	CopyNumToByte
INT	WORD	CopyNumToWord
DINT	DWORD	CopyNumToDword
LINT	LWORD	CopyNumToLword

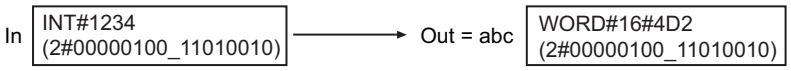
Ниже показан пример для команды CopyNumToWord, в котором *In* = INT#1234.

LD



ST

```
abc:=CopyNumToWord(INT#1234);
```



CopyNumTo** (Целое число со знаком в вещественное число)

Команда CopyNumTo** копирует содержимое целого числа со знаком непосредственно в вещественное число.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CopyNumTo**	Группа копирования комбинации битов (Целое число со знаком в вещественное число)	FUN	(@)CopyNumToReal EN ENO In Out (@)CopyNumToLreal EN ENO In Out	Out:=CopyNumToReal(In); или Out:=CopyNumToLreal(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник копирования	Вход	Источник копирования	Зависит от типа данных.	---	0
Out	Адресат копирования	Выход	Адресат копирования	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In												OK	OK							
Out	Если *In* — значение типа DINT, то должен использоваться тип REAL, а если LINT, то LREAL.																			

Функция

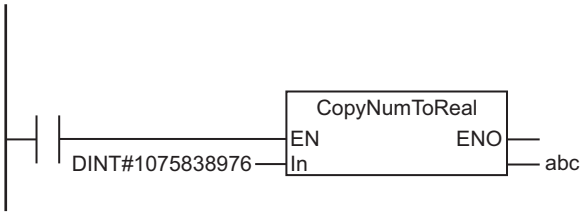
Группа команд CopyNumTo** копирует содержимое *In* (источник копирования) непосредственно в *Out* (адресат копирования).

В группу входят две команды для разных комбинаций типов данных *In* и *Out*, которые перечислены ниже.

In	Out	Команда
DINT	REAL	CopyNumToReal
LINT	LREAL	CopyNumToLreal

Ниже показан пример для команды CopyNumToReal, в котором *In* = DINT#1075838976.

LD



ST

```
abc:=CopyNumToReal(DINT#1075838976);
```

In	Out = abc
DINT#1075838976 (2#01000000_00100000_00000000_00000000)	REAL#2.5 (2#01000000_00100000_00000000_00000000)

Copy**To*** (Вещественное число в битовую строку)

Команда Copy**To*** копирует содержимое вещественного числа непосредственно в битовую строку.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Copy**To***	Группа копирования комбинации битов (Вещественное число в битовую строку)	FUN	(@)CopyRealToDword EN ENO In Out (@)CopyLrealToLword EN ENO In Out	Out:=CopyRealToDword(In); или Out:=CopyLrealToLword(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник копирования	Вход	Источник копирования	Зависит от типа данных.	---	0,0
Out	Адресат копирования	Выход	Адресат копирования	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														OK	OK					
Out	Если *In* — значение типа REAL, то должен использоваться тип DWORD, а если LREAL, то LWORD.																			

Функция

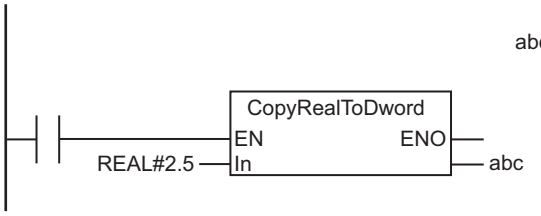
Группа команд Copy**To*** копирует содержимое *In* (источник копирования) непосредственно в *Out* (адресат копирования).

В группу входят две команды для разных комбинаций типов данных *In* и *Out*, которые перечислены ниже.

In	Out	Команда
REAL	DWORD	CopyRealToDword
LREAL	LWORD	CopyLrealToLword

Ниже показан пример для команды CopyRealToDword, в котором *In* = REAL#2.5.

LD



ST

```
abc:=CopyRealToDword(REAL#2.5);
```

In	Out = abc
REAL#2.5 (2#01000000_00100000_00000000_00000000)	DWORD#16#40200000 (2#01000000_00100000_00000000_00000000)

Copy**ToNum (Вещественное число в целое число со знаком)

Команда Copy**ToNum копирует содержимое вещественного числа непосредственно в целое число со знаком.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Copy**ToNum	Группа копирования комбинации битов (Вещественное число в целое число со знаком)	FUN	(@)CopyRealToNum EN ENO In Out (@)CopyLrealToNum EN ENO In Out	Out:=CopyRealToNum(In); или Out:=CopyLrealToNum(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник копирования	Вход	Источник копирования	Зависит от типа данных.	---	0,0
Out	Адресат копирования	Выход	Адресат копирования	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														OK	OK					
Out	Если *In* — значение типа REAL, то должен использоваться тип DINT, а если LREAL, то LINT.																			

Функция

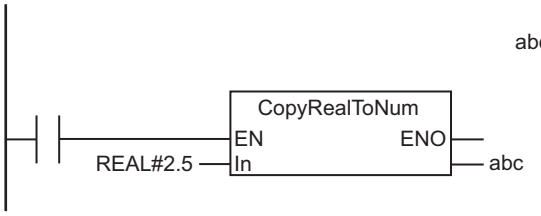
Группа команд Copy**ToNum копирует содержимое *In* (источник копирования) непосредственно в *Out* (адресат копирования).

В группу входят две команды для разных комбинаций типов данных *In* и *Out*, которые перечислены ниже.

In	Out	Команда
REAL	DINT	CopyRealToNum
LREAL	LINT	CopyLrealToNum

Ниже показан пример для команды CopyRealToNum, в котором *In* = REAL#2.5.

LD



ST

```
abc:=CopyRealToNum(REAL#2.5);
```

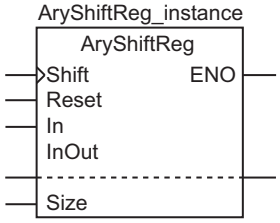
In	REAL#2.5 (2#01000000_00100000_00000000_00000000)	Out = abc	DINT#1075838976 (2#01000000_00100000_00000000_00000000)

Команды сдвига

Команда	Имя	Стр.
AryShiftReg	Регистр сдвига	стр. 2-442
AryShiftRegLR	Реверсивный регистр сдвига	стр. 2-445
ArySHL и ArySHR	Сдвиг массива влево на N элементов/Сдвиг массива вправо на N элементов	стр. 2-448
SHL и SHR	Сдвиг влево на N битов/Сдвиг вправо на N битов	стр. 2-451
NSHLC и NSHRC	Сдвиг на N битов влево с переносом/Сдвиг на N битов вправо с переносом	стр. 2-454
ROL и ROR	Циклический сдвиг на N битов влево/Циклический сдвиг на N битов вправо	стр. 2-457

AryShiftReg

Команда AryShiftReg сдвигает массив битовых строк на один бит влево и вставляет входное значение в самый младший значащий бит.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryShiftReg	Регистр сдвига	FB		AryShiftReg_instance(Shift, Reset, In, InOut, Size);

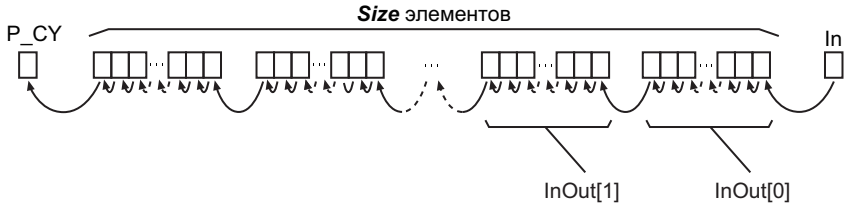
Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Shift	Сдвиг	Вход	Сдвиг, когда сигнал переходит в состояние ИСТИНА.	Зависит от типа данных.	---	ЛОЖЬ
Reset	Сброс		ИСТИНА: регистр сбрасывается.			
In	Входное значение		Значение для вставки в младший значащий бит InOut[].			
Size	Количество элементов в массиве битовых строк		Количество элементов для использования в качестве регистра сдвига в InOut[].			1
InOut[] (массив)	Массив битовых строк	Вход-выход	Массив битовых строк	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Shift	ОК																			
Reset	ОК																			
In	ОК																			
Size							ОК													
InOut[] (массив)	ОК	ОК	ОК	ОК	ОК															

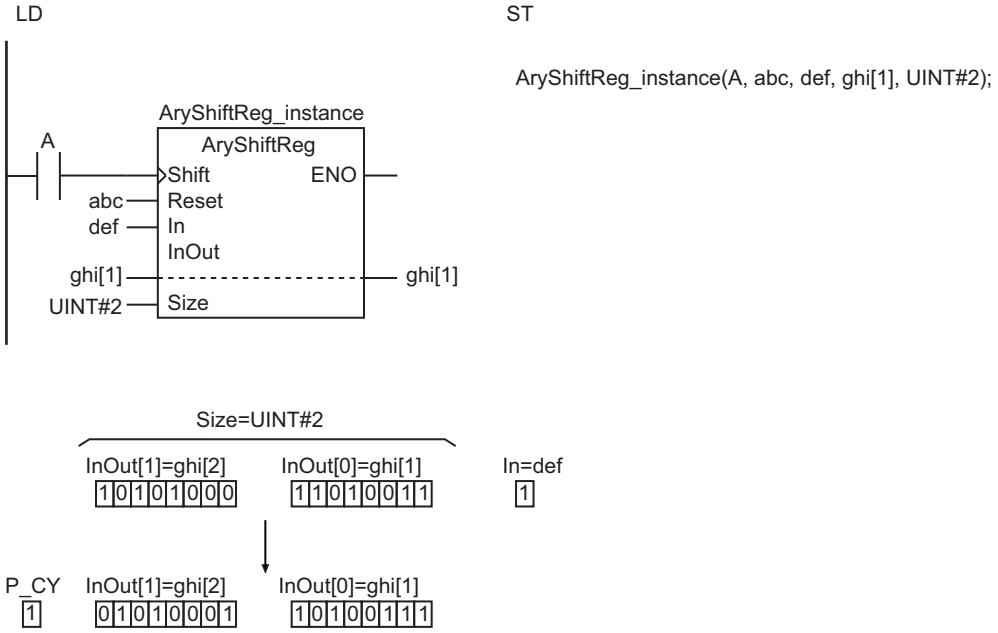
Функция

Команда AryShiftReg сдвигает *Size* элементов массива битовых строк InOut[] на один бит влево (т. е. в сторону самого старшего бита), начиная с элемента InOut[0], когда состояние входа *Shift* меняется на ИСТИНА.
В младший значащий бит вставляется значение *In* (входное значение). Содержимое самого старшего значащего бита, который оказывается сдвинут за пределы массива битовых строк, выводится во флаг переноса (CY) (*P_CY*).



Когда *Reset* = ИСТИНА, флаг CY и все биты в *Size* элементах, начиная с InOut[0], сбрасываются в ЛОЖЬ.

В показанном ниже примере программы InOut[] является массивом значений типа BYTE, а *Size* = UINT#2.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
P_CY	Флаг переноса (CY)	BOOL	Значение, сохраненное во флаг переноса

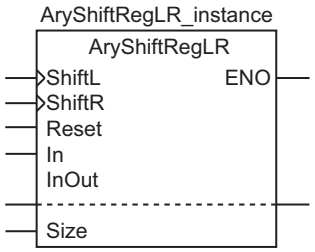
Меры предосторожности для обеспечения надлежащей эксплуатации

- Пока вход *Reset* = ИСТИНА, регистр не сдвигается, даже если вход *Shift* переходит в состояние ИСТИНА.

- Состояние *ENO* поменяется на ИСТИНА, если вход *Shift* перейдет в состояние ИСТИНА и будет нормально выполнена операция сдвига, либо если вход *Reset* перейдет в состояние ИСТИНА и будет нормально выполнена операция сброса.
- Значение выхода InOut[] не изменяется, если *Size = 0*.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое InOut[] не изменится.
  a) Значение *Size* приводит к выходу за область массива InOut[].

AryShiftRegLR

Команда AryShiftRegLR сдвигает массив битовых строк на один бит влево или вправо и вставляет входное значение в самый младший или самый старший значащий бит.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryShiftRegLR	Реверсивный регистр сдвига	FB		AryShiftRegLR_instance(ShiftL, ShiftR, Reset, In, InOut, Size);

Переменные

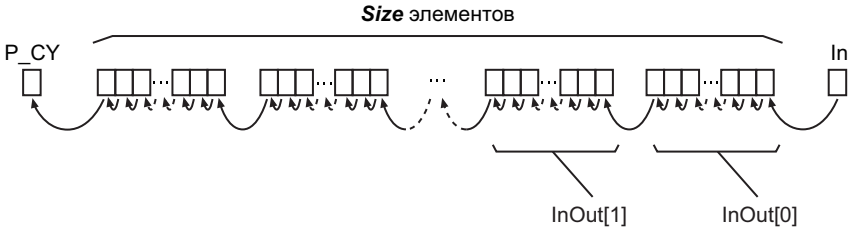
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
ShiftL	Сдвиг влево	Вход	Сдвиг влево, когда сигнал переходит в состояние ИСТИНА.	Зависит от типа данных.	---	ЛОЖЬ
ShiftR	Сдвиг вправо		Сдвиг вправо, когда сигнал переходит в состояние ИСТИНА.			
Reset	Сброс		ИСТИНА: регистр сбрасывается.			
In	Входное значение		Значение для вставки в младший или старший значащий бит InOut[].			
Size	Количество элементов в массиве битовых строк		Количество элементов для использования в качестве регистра сдвига в InOut[].			1
InOut[] (массив)	Массив битовых строк	Вход-выход	Массив битовых строк	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ShiftL	OK																			
ShiftR	OK																			
Reset	OK																			
In	OK																			

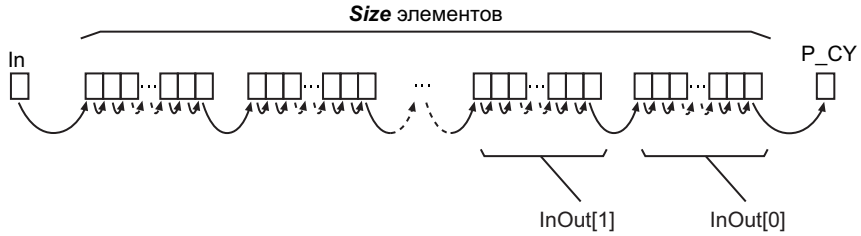
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size							ОК													
InOut[] (массив)	ОК	ОК	ОК	ОК	ОК															

Функция

Команда AryShiftRegLR сдвигает *Size* элементов массива битовых строк InOut[] на один бит влево, начиная с элемента InOut[0], когда состояние входа *ShiftL* меняется на ИСТИНА.
В младший значащий бит вставляется значение *In* (входное значение).
Содержимое самого старшего значащего бита, который оказывается сдвинут за пределы массива битовых строк, выводится во флаг переноса (CY) (*P_CY*).

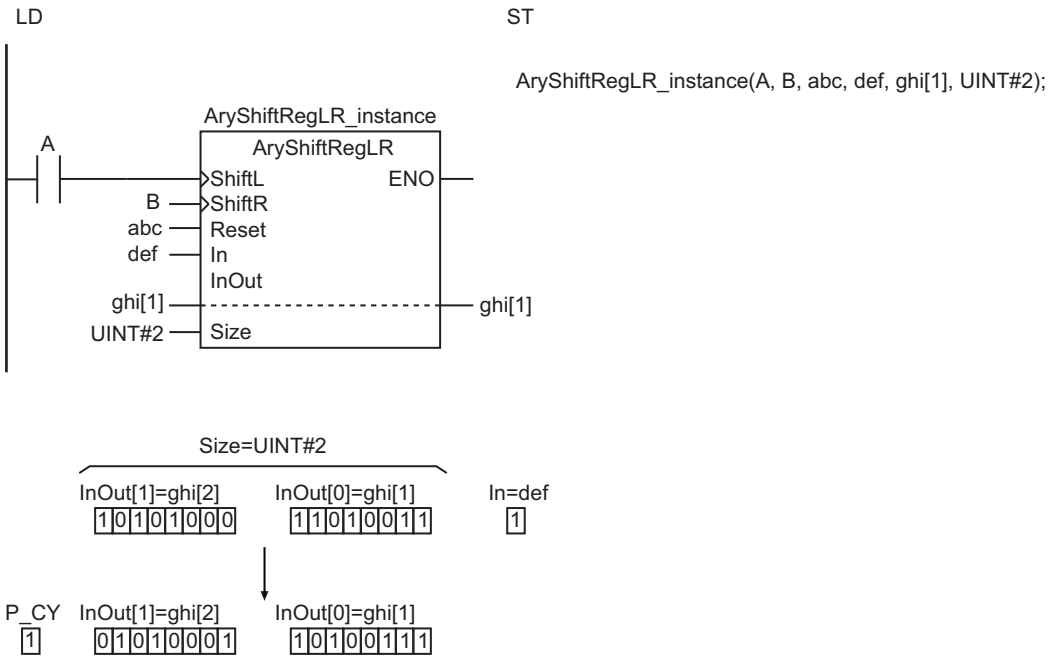


Когда состояние входа *ShiftR* меняется на ИСТИНА, биты сдвигаются на один бит вправо, а в старший значащий бит вставляется значение *In*.
Содержимое самого младшего значащего бита, который оказывается сдвинут за пределы массива битовых строк, выводится во флаг переноса (CY) (*P_CY*).



Когда *Reset* = ИСТИНА, флаг *P_CY* и все биты в *Size* элементах, начиная с InOut[0], сбрасываются в ЛОЖЬ.

В показанном ниже примере программы *InOut* является массивом значений типа BYTE, *Size* = UINT#2, в состояние ИСТИНА переходит вход *ShiftL*.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
P_CY	Флаг переноса (CY)	BOOL	Значение, сохраненное во флаг переноса

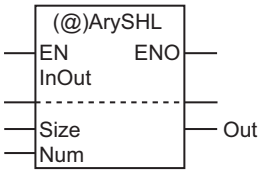
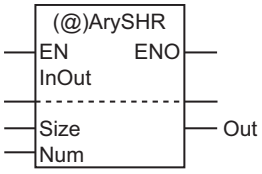
Меры предосторожности для обеспечения надлежащей эксплуатации

- Пока вход *Reset* = ИСТИНА, регистр не сдвигается, даже если вход *ShiftL* или *ShiftR* переходит в состояние ИСТИНА.
- Если входы *ShiftL* и *ShiftR* переходят в состояние ИСТИНА одновременно, регистр не сдвигается.
- Состояние *ENO* поменяется на ИСТИНА, если вход *ShiftL* или *ShiftR* перейдет в состояние ИСТИНА и будет нормально выполнена операция сдвига, либо если вход *Reset* перейдет в состояние ИСТИНА и будет нормально выполнена операция сброса.
- Значение выхода InOut[] не изменяется, если *Size = 0*.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое InOut[] не изменится.
  a) Значение *Size* приводит к выходу за область массива InOut[].

ArySHL и ArySHR

Эти команды сдвигают элементы массива на один или несколько элементов.

ArySHL : Сдвигает массив влево (в сторону элементов с большими индексами).
ArySHR : Сдвигает массив вправо (в сторону элементов с меньшими индексами).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ArySHL	Сдвиг массива влево на N элементов	FUN		ArySHL(InOut, Size, Num);
ArySHR	Сдвиг массива вправо на N элементов	FUN		ArySHR(InOut, Size, Num);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Size	Количество элементов в регистре сдвига	Вход	Количество элементов в регистре сдвига	Зависит от типа данных.	---	1
Num	Количество сдвигаемых элементов		Количество сдвигаемых элементов			
InOut[] (массив)	Массив регистра сдвига	Вход-выход	Массив регистра сдвига	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size							ОК													
Num							ОК													
InOut[] (массив)	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также допускается указывать массивы структур.																			
Out	ОК																			

Функция

Эти команды сдвигают *Size* старших элементов массива InOut[] (массив регистра сдвига) на *Num* элементов.
Значения, которые оказываются сдвинуты за пределы массива, отбрасываются.
В пустые элементы записывается начальное значение по умолчанию, соответствующее типу данных InOut[].
Если InOut[] является массивом структур, то инициализируются все члены этих структур.

Ниже приведены значения по умолчанию для разных типов данных.

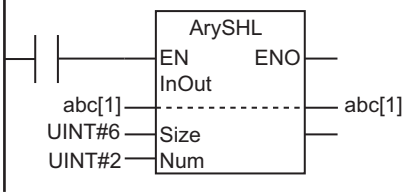
Тип данных	По умолчанию
BOOL	ЛОЖЬ
BYTE, WORD, DWORD или LWORD	16#0
USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL или LREAL	0
TIME	T#0ms
DATE	D#1970-1-1
TOD	TOD#0:0:0
DT	DT#1970-1-1-0:0:0
STRING	''

ArySHL

Команда ArySHL сдвигает массив влево (в сторону элементов с большими индексами).

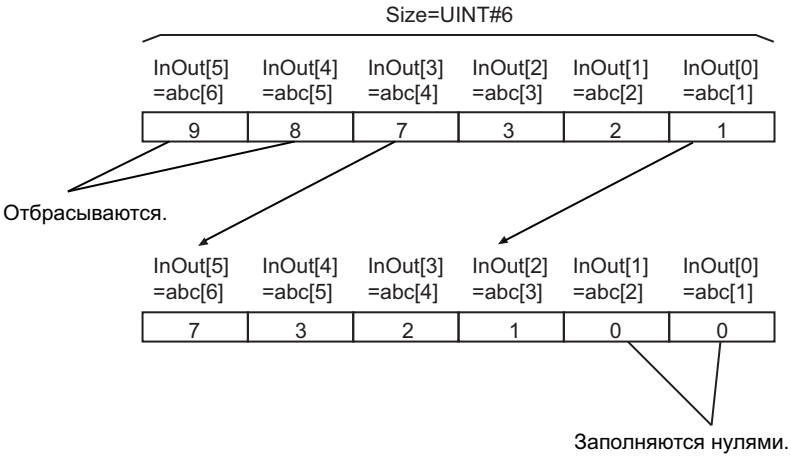
Ниже приведен пример использования команды ArySHL, в котором *Size* = UINT#6, а *Num* = UINT#2.

LD



ST

```
ArySHL(abc[1], UINT#6, UINT#2);
```



ArySHR

Команда ArySHR сдвигает массив вправо (в сторону элементов с меньшими индексами).

Дополнительная информация

Если InOut[] является массивом типа BOOL, результат будет таким же, как при сдвиге битовой строки из *Size* битов на *Num* битов.

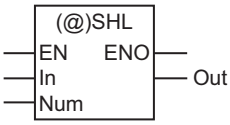
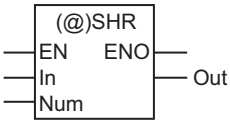
Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *Num* = *0*, то операция сдвига не выполняется.
- Если значение *Num* больше значения *Size*, то все значения от InOut[0] до InOut[*Size*-1] инициализируются.
- При использовании этих команд в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое InOut[] не изменится.
  a) Значение *Size* приводит к выходу за область массива InOut[].

SHL и SHR

Эти команды сдвигают битовую строку на один или несколько битов.

SHL : Сдвигает битовую строку влево (в сторону старших битов).
SHR : Сдвигает битовую строку вправо (в сторону младших битов).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SHL	Сдвиг влево на N битов	FUN		Out:=SHL(In, Num);
SHR	Сдвиг вправо на N битов	FUN		Out:=SHR(In, Num);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для сдвига	Вход	Данные для сдвига	Зависит от типа данных.	---	*1
Num *2	Количество для сдвига		Количество сдвигаемых битов	От 0 до количества битов в *In*	Биты	1
Out	Результат обработки	Выход	Результат обработки	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

*2. В Sysmac Studio версии 1.03 или более поздней версии есть возможность использовать *N* вместо *Num*. Так нагляднее отображается соответствие между переменными и именами параметров в выражениях языка ST. Например, можно использовать следующую форму записи: Out:=SHL(In:=BYTE#16#89, N:=ULINT#2);.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Num						OK *1			OK *1											
Out	Тип данных должен быть таким же, как у *In*.																			

*1. Если используются модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше, используйте переменную типа ULINT. В случае модуля ЦПУ с версией модуля 1.01 или более ранней и Sysmac Studio версии 1.02 или ниже используйте переменную типа USINT.

Функция

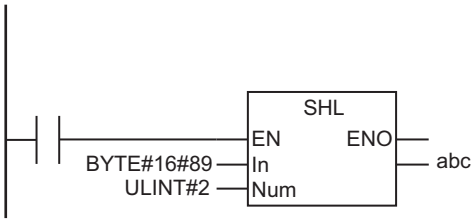
Эти команды сдвигают содержимое битовой строки *In* (данные для сдвига) на количество битов, указанное в *Num* (количество для сдвига).
Биты, которые оказываются сдвинуты за пределы строки, отбрасываются, а в позиции сдвинутых битов в противоположном конце регистра вставляются нули.

SHL

Команда SHL сдвигает биты справа налево (от младших битов к старшим).

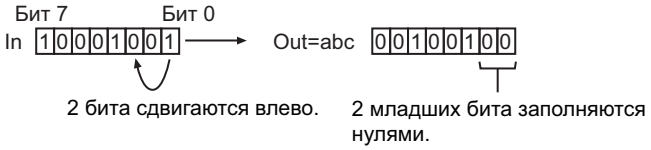
Ниже показан пример для случая, когда *In* = BYTE#16#89, а *Num* = ULINT#2.

LD



ST

```
abc:=SHL(BYTE#16#89, ULINT#2);
```

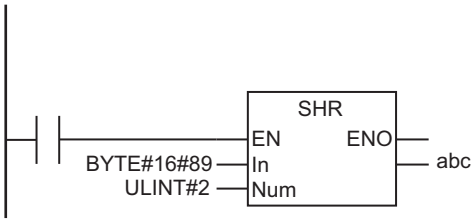


SHR

Команда SHR сдвигает биты слева направо (от старших битов к младшим).

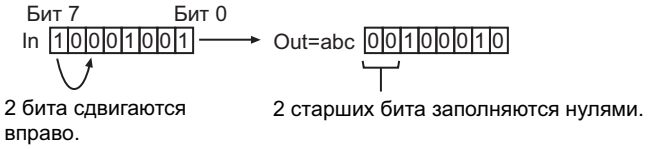
Ниже показан пример для случая, когда *In* = BYTE#16#89, а *Num* = ULINT#2.

LD



ST

```
abc:=SHR(BYTE#16#89, ULINT#2);
```



Дополнительная информация

Если нужно, чтобы биты, которые вытесняются из регистра, вставлялись в противоположном конце регистра, используйте команды ROL и ROR.

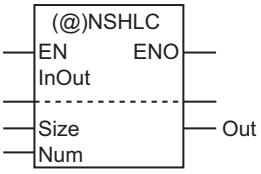
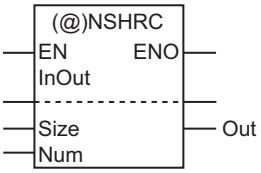
Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных *In* и *Out* должны быть одинаковыми.
- Если *Num* = *0*, ошибки не произойдет. В этом случае переменной *Out* будет непосредственно присвоено значение переменной *In*.
- Если значение *Num* превышает количество битов, указанное в *In*, ошибки не произойдет. Переменной *Out* будет присвоено значение 16#0.

NSHLC и NSHRC

Эти команды сдвигают массив битовых строк на один или несколько битов с использованием флага переноса (CY).

NSHLC : Сдвигает массив влево (в сторону элементов с большими индексами).
NSHRC : Сдвигает массив вправо (в сторону элементов с меньшими индексами).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NSHLC	Сдвиг на N битов влево с переносом	FUN		NSHLC(InOut, Size, Num);
NSHRC	Сдвиг на N битов вправо с переносом	FUN		NSHRC(InOut, Size, Num);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Size	Количество битов в регистре сдвига	Вход	Количество битов в регистре сдвига	Зависит от типа данных.	Биты	1
Num	Количество сдвигаемых битов		Количество сдвигаемых битов			
InOut[] (массив)	Массив регистра сдвига	Вход-выход	Массив битовых строк для сдвига	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size						OK														
Num						OK														
InOut[] (массив)	OK	OK	OK	OK	OK															
Out	OK																			

Функция

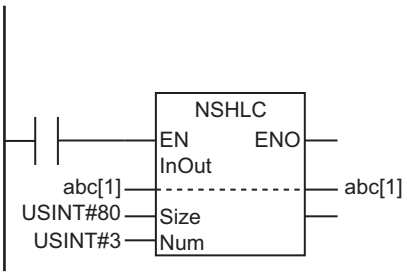
Эти команды сдвигают *Size* элементов массива InOut[] (массив регистра сдвига) на количество битов, указанное в *Num*. Регистр сдвига начинается с элемента InOut[0].
Содержимое последнего бита, который оказывается сдвинут за пределы регистра, выводится во флаг переноса (CY). В позиции сдвинутых битов в противоположном конце регистра вставляются нули.

NSHLC

Команда NSHLC сдвигает биты в направлении от элементов с меньшими индексами к элементам с большими индексами и от младших битов к старшим.

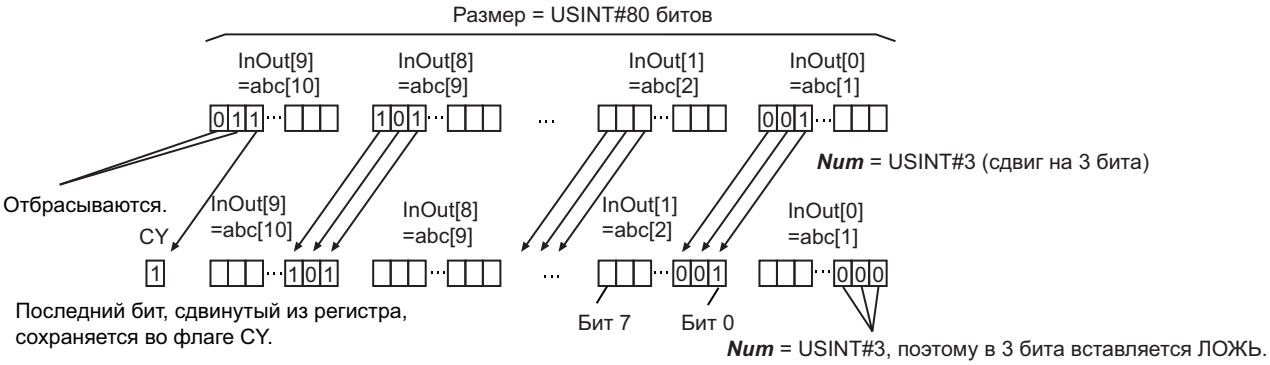
Пример ниже демонстрирует работу команды NSHLC для случая, когда InOut[] является массивом значений типа BYTE, *Size* = USINT#80, а *Num* = USINT#3.

LD



ST

```
NSHLC(abc[1], USINT#80, USINT#3);
```



NSHRC

Команда NSHRC сдвигает биты в направлении от элементов с большими индексами к элементам с меньшими индексами и от старших битов к младшим.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
P_CY	Флаг переноса (CY)	BOOL	Значение, сохраненное во флаг переноса

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *Num* = *0*, то операция сдвига не выполняется.
- Если значение *Num* больше значения *Size*, то *Size* битов, начиная с бита 0 элемента InOut[0], сбрасываются в ЛОЖЬ. Значение флага переноса (CY) меняется на ЛОЖЬ.
- При использовании этих команд в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое InOut[] не изменится.
  a) Значение *Size* приводит к выходу за область массива InOut[].

ROL и ROR

Эти команды циклически сдвигают битовую строку на один или несколько битов.

ROL : Циклически сдвигает битовую строку влево (в сторону старших битов).
ROR : Циклически сдвигает битовую строку вправо (в сторону младших битов).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ROL	Циклический сдвиг на N битов влево	FUN	(@)ROL EN ENO In Out Num	Out:=ROL(In, Num);
ROR	Циклический сдвиг на N битов вправо	FUN	(@)ROR EN ENO In Out Num	Out:=ROR(In, Num);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для циклического сдвига	Вход	Данные для циклического сдвига	Зависит от типа данных.	---	*1
Num*2	Количество битов		Количество циклически сдвигаемых битов	От 0 до количества битов в *In*	Биты	1
Out	Результат обработки	Выход	Результат обработки	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.
*2. В Sysmac Studio версии 1.03 или более поздней версии есть возможность использовать *N* вместо *Num*. Так нагляднее отображается соответствие между переменными и именами параметров в выражениях языка ST. Например, можно использовать следующую форму записи: Out:=ROL(In:=BYTE#16#89, N:=ULINT#2);.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Num						OK *1			OK *1											
Out	Тип данных должен быть таким же, как у *In*.																			

*1. Если используются модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше, используйте переменную типа ULINT. В случае модуля ЦПУ с версией модуля 1.01 или более ранней и Sysmac Studio версии 1.02 или ниже используйте переменную типа USINT.

Функция

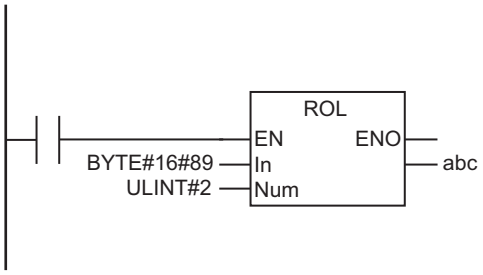
Эти команды циклически сдвигают содержимое битовой строки *In* (данные для циклического сдвига) на количество битов, указанное в *Num* (количество битов). Биты, которые оказываются сдвинуты за пределы регистра, вставляются в противоположный конец регистра.

ROL

Команда ROL циклически сдвигает биты справа налево (в направлении от младших битов к старшим).

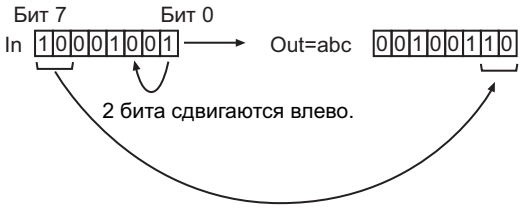
Ниже показан пример для случая, когда *In* = BYTE#16#89, а *Num* = ULINT#2.

LD



ST

```
abc:=ROL(BYTE#16#89, ULINT#2);
```



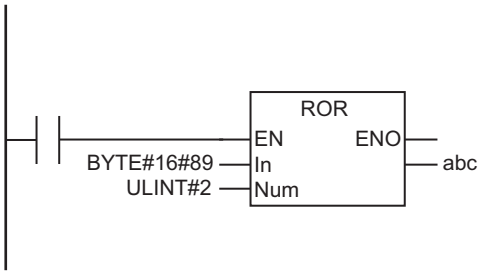
2 старших бита ***In*** вставляются в 2 младших бита.

ROR

Команда ROR циклически сдвигает биты слева направо (в направлении от старших битов к младшим).

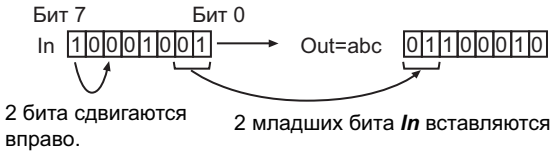
Ниже показан пример для случая, когда *In* = BYTE#16#89, а *Num* = ULINT#2.

LD



ST

```
abc:=ROR(BYTE#16#89, ULINT#2);
```



Дополнительная информация

Если нужно, чтобы биты, вытесняемые из регистра, отбрасывались, а в позиции сдвинутых битов в противоположном конце регистра вставлялись нули, используйте команды SHL и SHR.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных *In* и *Out* должны быть одинаковыми.
- Если *Num = 0*, ошибки не произойдет. В этом случае переменной *Out* будет непосредственно присвоено значение переменной *In*.
- Если значение *Num* превышает количество битов, указанное в *In*, ошибки не произойдет. Биты будут циклически сдвинуты на количество битов, указанное в *Num*. Например, если *In* содержит значение типа WORD, то при значениях USINT#1 и USINT#17 в переменной *Num* значение *Out* будет одним и тем же.

Команды преобразования

Команда	Имя	Стр.
Swap	Перестановка байтов	стр. 2-463
Neg	Инверсия знака	стр. 2-465
Decoder	Битовый дешифратор	стр. 2-467
Encoder	Битовый шифратор	стр. 2-470
BitCnt	Подсчет битов	стр. 2-472
ColmToLine_**	Группа преобразования столбца в строку	стр. 2-474
LineToColm	Преобразование строки в столбец	стр. 2-476
Gray	Преобразование кода Грея	стр. 2-479
UTF8ToSJIS	Преобразование кодов символов (из UTF-8 в SJIS)	стр. 2-484
SJISToUTF8	Преобразование кодов символов (из SJIS в UTF-8)	стр. 2-486
PWLApprox и PWLApproxNoLineChk	Аппроксимация ломаной линии с проверкой данных ломаной линии/ Аппроксимация ломаной линии без проверки данных ломаной линии	стр. 2-488
PWLLineChk	Проверка данных ломаной линии	стр. 2-494
MovingAverage	Скользящее среднее	стр. 2-497
DispartReal	Разделение на мантиссу и показатель степени	стр. 2-504
UniteReal	Формирование вещественного числа из мантиссы и показателя степени	стр. 2-507
NumToDecString и NumToHexString	Преобразование в текстовую строку фиксированной длины в десятичном виде/Преобразование в текстовую строку фиксированной длины в шестнадцатеричном виде	стр. 2-509
HexStringToNum_**	Группа преобразования текстовой строки с шестнадцатеричными цифрами в число	стр. 2-513
FixNumToString	Преобразование числа с фиксированной запятой в текстовую строку	стр. 2-515
StringToFixNum	Преобразование текстовой строки в число с фиксированной запятой	стр. 2-517
DtToString	Преобразование даты и времени в текстовую строку	стр. 2-520

Команда	Имя	Стр.
DateToString	Преобразование даты в текстовую строку	стр. 2-522
TodToString	Преобразование времени суток в текстовую строку	стр. 2-524
GrayToBin_** и BinToGray_**	Группа преобразования кода Грея в двоичный код/Преобразование двоичного кода в код Грея	стр. 2-526
StringToAry	Преобразование текстовой строки в массив	стр. 2-529
AryToString	Преобразование массива в текстовую строку	стр. 2-531
DispartDigit	Разделение на 4-битные блоки	стр. 2-533
UniteDigit_**	Группа объединения 4-битных блоков	стр. 2-535
Dispart8Bit	Разделение данных на байты	стр. 2-537
Unite8Bit_**	Группа объединения байтов данных	стр. 2-539
ToAryByte	Преобразование в байтовый массив	стр. 2-541
AryByteTo	Преобразование из байтового массива	стр. 2-547
SizeOfAry	Получить количество элементов массива	стр. 2-554
PackWord	Объединение двух байтов	стр. 2-556
PackDword	Объединение четырех байтов	стр. 2-558
LOWER_BOUND и UPPER_BOUND	Получить минимальное значение индекса массива/Получить максимальное значение индекса массива	стр. 2-560

Swap

Команда Swap меняет местами старший и младший байты 16-битного значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Swap	Перестановка байтов	FUN	(@)Swap EN ENO In Out	Out:=Swap(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	0
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

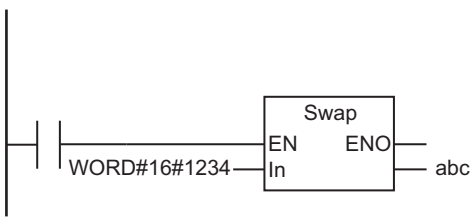
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In			OK																	
Out			OK																	

Функция

Команда Swap меняет местами старший и младший байты переменной *In* (данные для преобразования) и присваивает результат переменной *Out* (результат преобразования).

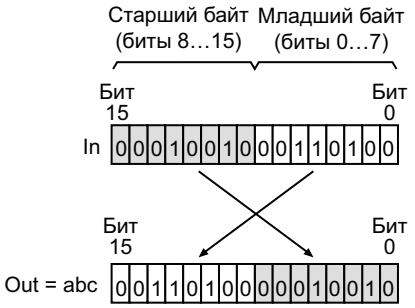
В показанном ниже примере программы *In* = WORD#16#1234.

LD



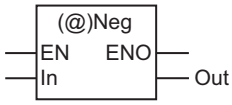
ST

```
abc:=Swap(WORD#16#1234);
```


Старший байт
(биты 8…15)
Младший байт
(биты 0…7)
Бит
15
Бит
0
In
0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0
Бит
15
Бит
0
Out = abc
0 0 1 1 0 1 0 0 0 0 0 1 0 0 1 0

Neg

Команда Neg меняет знак числа на противоположный.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Neg	Инверсия знака	FUN		Out:=Neg(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	*1
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
Out						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					

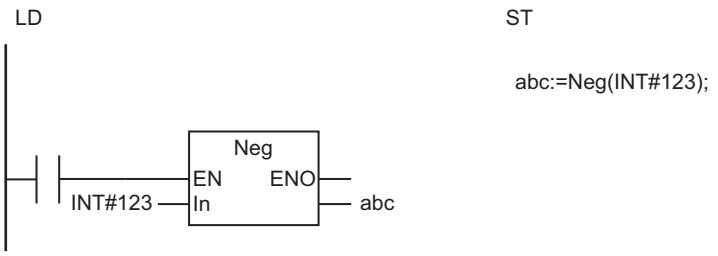
Функция

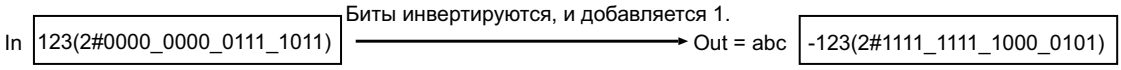
Команда Neg меняет знак значения переменной *In* (данные для преобразования) на противоположный.

Обработка значения зависит от типа данных *In*, что показано ниже.

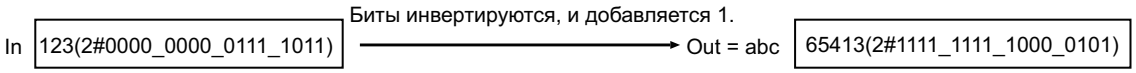
Тип данных *In*	Значение *Out*
Целое число знаком: SINT, INT, DINT или LINT	Все биты в *In* инвертируются, а затем добавляется 1 (это эквивалентно умножению *In* на -1).
Целые числа без знака: USINT, UNIT, UDINT или ULINT	Все биты в *In* инвертируются, а затем добавляется 1
Вещественные числа: REAL или LREAL	*In* × (-1)

В показанном ниже примере программы *In* = INT#123.



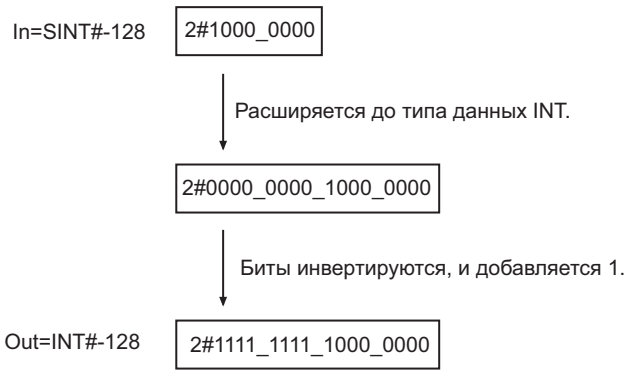


В показанном ниже примере программы *In* = UINT#123.



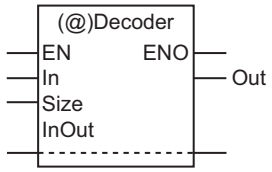
Меры предосторожности для обеспечения надлежащей эксплуатации

Если для переменных *In* и *Out* используются разные типы данных, проследите за тем, чтобы диапазон допустимых значений *Out* вмещал в себя диапазон допустимых значений *In*. Если это условие не будет соблюдено, ошибки не произойдет, однако *Out* будет содержать недопустимое значение.
Например, если *In* = SINT#-128, а *Out* относится к типу данных INT, то *Out* будет равно INT#-128, а не INT#128.



Decoder

Команда Decoder устанавливает указанный бит в состояние ИСТИНА, а остальные биты сбрасывает в состояние ЛОЖЬ в массиве элементов, состоящем максимум из 256 битов.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Decoder	Битовый дешифратор	FUN		Decoder(In, Size, InOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Положение преобразуемого бита	Вход	Позиция бита для преобразования	Зависит от типа данных.	---	0
Size	Биты для преобразования		Количество битов для преобразования	0...8	Биты	1
InOut[] (массив)	Массив для преобразования	Вход-выход	Массив для преобразования	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		ОК																		
Size						ОК														
InOut[] (массив)	ОК	ОК	ОК	ОК	ОК															
Out	ОК																			

Функция

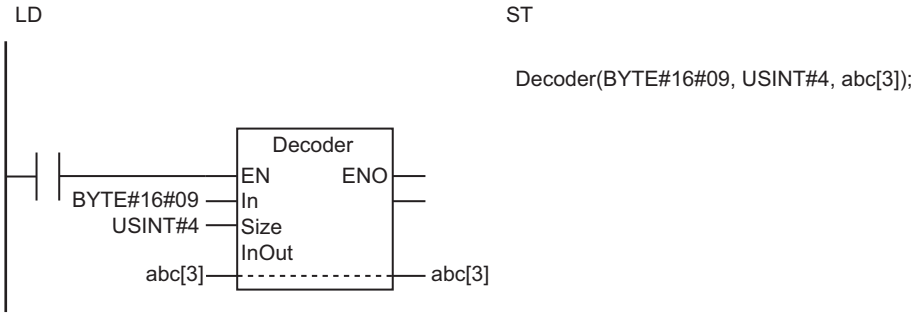
Команда Decoder обрабатывает 2^{Size} битов массива InOut[], который начинается с элемента InOut[0], и переводит указанный бит в состояние ИСТИНА. Все остальные биты сбрасываются в состояние ЛОЖЬ.

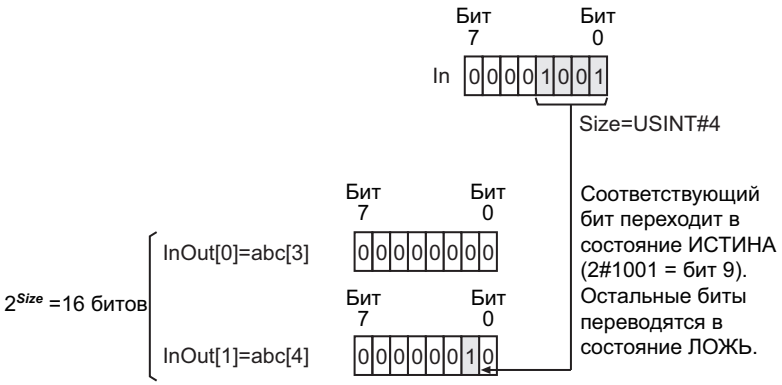
Позиция бита, переводимого в состояние ИСТИНА, определяется параметрами *Size* и *In* (положение преобразуемого бита).

Всегда присоединяйте номер элемента к входному-выходному параметру, который передается в переменную InOut[], например: array[3].

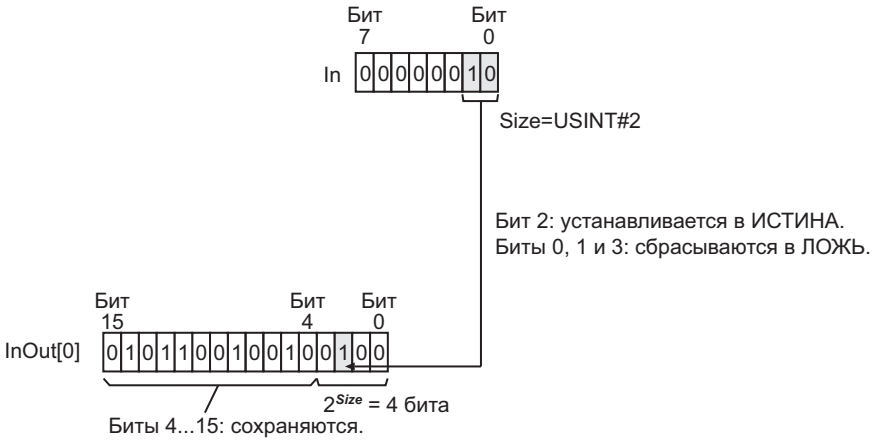
Рассмотрим пример, в котором *In* = BYTE#16#09, *Size* = USINT#4, а InOut[] — массив типа BYTE. Так как *In* (положение преобразуемого бита) содержит значение 16#09 (т. е. 9 в десятичной записи), в состояние ИСТИНА переводится девятый по счету младший бит в InOut[], а все остальные биты сбрасываются в состояние ЛОЖЬ.

InOut[] — это массив значений типа BYTE, поэтому девятым по счету битом относительно самого младшего бита будет бит 1 в InOut[1]. Следовательно, в состояние ИСТИНА устанавливается бит 1 элемента InOut[1], а все остальные биты элемента InOut[1] и все биты элемента InOut[0] сбрасываются в ЛОЖЬ.





Если количество битов в элементах InOut[] больше, чем количество битов, заданное параметром *Size*, значения остальных битов остаются прежними. Рассмотрим пример, в котором *In* = BYTE#16#02, *Size* = USINT#2, а InOut[] — массив типа WORD.
Так как *Size* = USINT#2, обрабатываются 4 младших бита элемента InOut[0]. Значения остальных битов элемента InOut[0] (биты 4...15) остаются прежними.



Дополнительная информация

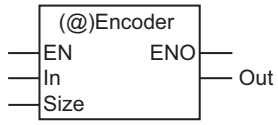
Для определения позиции самого старшего бита в состоянии ИСТИНА в элементах массива, содержащего максимум 256 битов, используйте команду *Encoder* на стр. 2-470.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *Size = 0*, то все биты в InOut[] переходят в состояние ЛОЖЬ.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое InOut[] не изменится.
  a) Значение *Size* находится за пределами допустимого диапазона.
  b) Значение 2^{Size} превышает количество битов в элементах массива InOut[].

Encoder

Команда Encoder определяет позицию самого старшего бита в состоянии ИСТИНА в элементах массива, содержащего максимум 256 битов.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Encoder	Битовый шифратор	FUN		Out:=Encoder(In, Size);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для преобразования	Вход	Массив для преобразования	Зависит от типа данных.	---	*1
Size	Биты для преобразования		Количество битов для преобразования	0...8	Биты	1
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)	ОК	ОК	ОК	ОК	ОК															
Size						ОК														
Out		ОК																		

Функция

Команда Encoder определяет позицию бита в состоянии ИСТИНА в указанной группе битов в массиве In[] (массив для преобразования).

Команда производит поиск бита, находящегося в состоянии ИСТИНА, среди 2^{Size} битов массива In[], начиная с элемента In[0]. Позиция бита в состоянии ИСТИНА в данной группе выражается в двоичном формате и сохраняется в младшие *Size* битов переменной *Out* (результат преобразования). Все остальные биты переменной *Out* сбрасываются в состояние ЛОЖЬ.

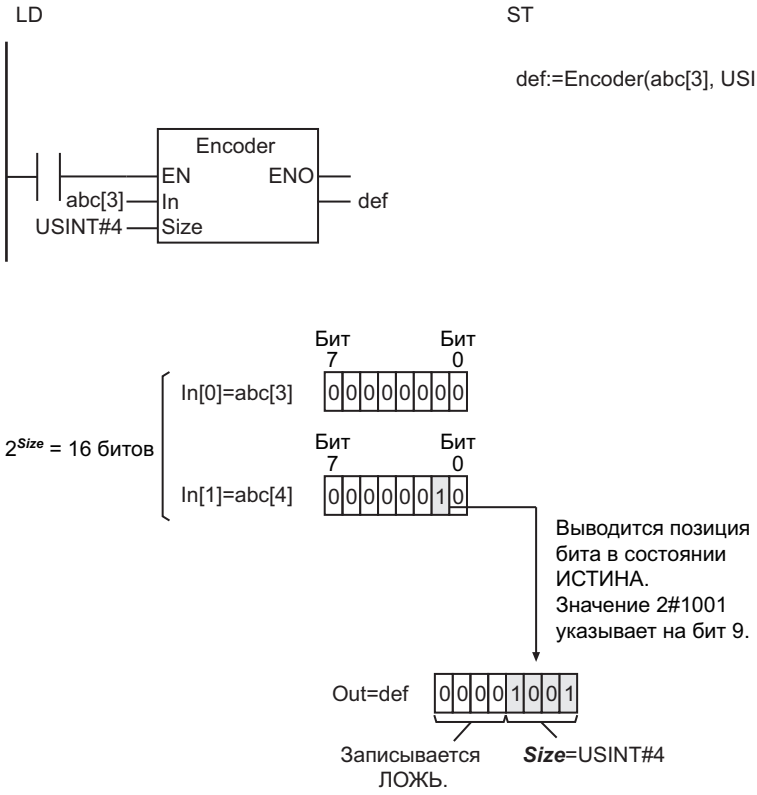
Если в указанной группе имеется несколько битов в состоянии ИСТИНА, команда определяет и выдает позицию самого старшего из этих битов.

Всегда присоединяйте номер элемента к входному параметру, который передается в In[], например: array[3].

Рассмотрим пример, в котором *Size* = USINT#4, а In[] — массив типа BYTE.

Так как *Size* = USINT#4, то поиск бита в состоянии ИСТИНА производится среди 16 (2^4) битов, начиная с In[0]. На следующем рисунке девятый по счету бит в группе находится в состоянии ИСТИНА.

Поскольку *Size* = USINT#4, то значение 2#1001 (т. е. 9 в десятичной записи) сохраняется в четырех младших битах переменной *Out*. Четыре старших бита переменной *Out* сбрасываются в состояние ЛОЖЬ.



Дополнительная информация

Для перевода только одного бита в состояние ИСТИНА и сброса остальных битов в состояние ЛОЖЬ в массиве элементов, состоящем максимум из 256 битов, используйте команду *Decoder* на стр. 2-467.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *Size* = *0*, то все биты в *Out* переходят в состояние ЛОЖЬ.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Size* находится за пределами допустимого диапазона.
  b) Значение 2^{Size} превышает количество битов в элементах массива In[].
  c) Все биты в In[], указанные параметром *Size*, находятся в состоянии ЛОЖЬ.

BitCnt

Команда BitCnt подсчитывает количество битов в состоянии ИСТИНА в битовой строке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
BitCnt	Подсчет битов	FUN	(@)BitCnt EN ENO In Out	Out:=BitCnt(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Строка для подсчета	Вход	Строка, в которой подсчитываются биты в состоянии ИСТИНА	Зависит от типа данных.	---	*1
Out	Результат подсчета	Выход	Количество битов в состоянии ИСТИНА	От 0 до количества битов в *In*	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

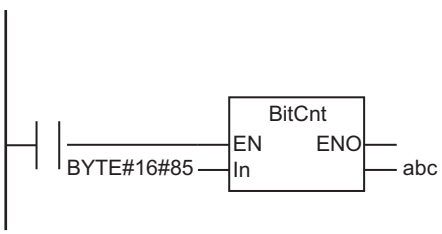
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Out						OK														

Функция

Команда BitCnt подсчитывает количество битов, находящихся в состоянии ИСТИНА, в *In*.

В представленном ниже примере переменная *In* содержит значение типа BYTE, которое равно BYTE#16#85.

LD



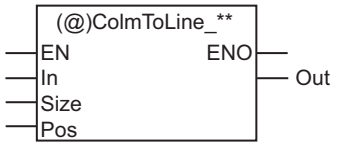
ST

```
abc:=BitCnt(BYTE#16#85);
```


In
1 0 0 0 1 0 0 1
Подсчитывается количество битов
в состоянии ИСТИНА (1).
Out = abc
3

ColmToLine_**

Команда ColmToLine_** извлекает значения битов в указанной позиции каждого элемента массива и выводит их в виде битовой строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ColmToLine_**	Группа преобразования столбца в строку	FUN	 "**" — тип данных, относящийся к битовым строкам.	Out:=ColmToLine_**(In, Size, Pos); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для преобразования	Вход	Массив для преобразования	Зависит от типа данных.	---	*1
Size	Количество элементов для преобразования		Количество элементов в In[] для преобразования	От 0 до количества битов в *Out*		1
Pos	Позиция бита для преобразования		Позиция бита для преобразования	От 0 до количества битов в In[] - 1		0
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)		OK	OK	OK	OK															
Size						OK														
Pos						OK														
Out		OK	OK	OK	OK															

Функция

Команда ColmToLine_** извлекает значения битов в указанных позициях элементов массива и выводит их по порядку в виде битовой строки.

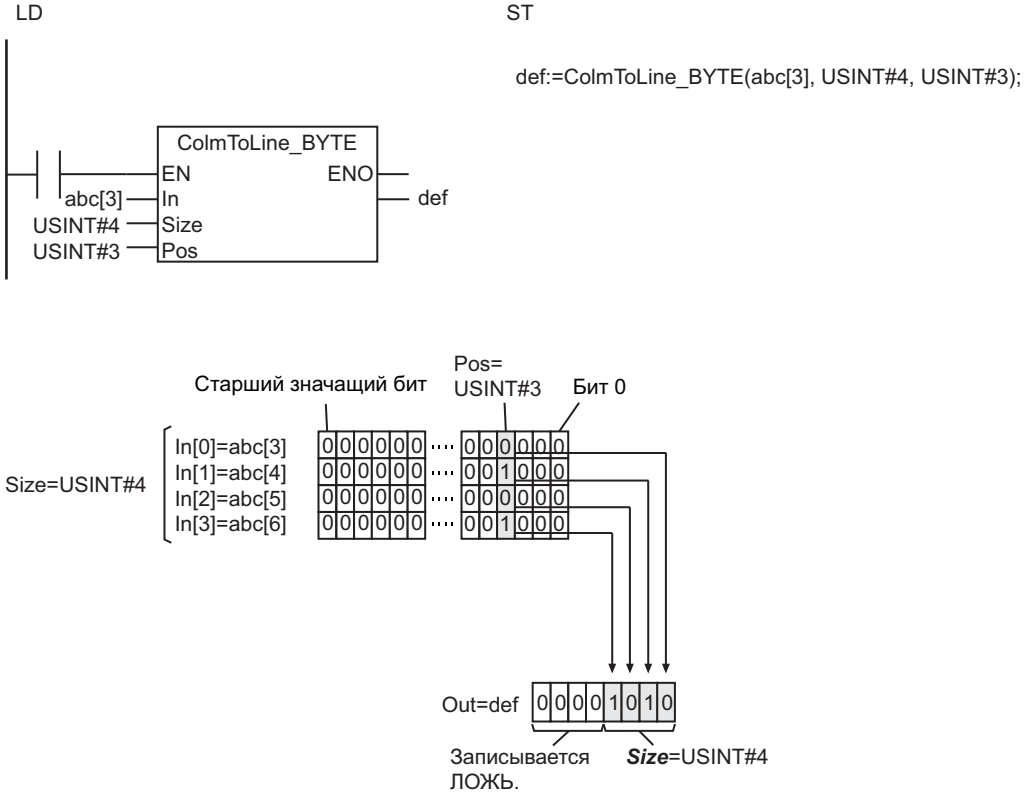
Сначала извлекаются *Size* элементов массива In[] (массив для преобразования), начиная с In[0].

Затем извлекается значение *Pos*-го бита каждого элемента.
Извлеченные значения преобразуются в битовую строку из *Size* битов, которая сохраняется в младшие биты *Out* (результат преобразования).
Остальные биты переменной *Out* сбрасываются в состояние ЛОЖЬ.

Имя команды определяется типом данных *Out*. Например, если *Out* относится к типу данных BYTE, команда будет иметь имя ColmToLine_BYTE.

Всегда присоединяйте номер элемента к входному параметру, который передается в In[], например: array[3].

Ниже приведен пример работы команды ColmToLine_BYTE для случая, когда *Pos* = USINT#3, а *Size* = USINT#4.



Дополнительная информация

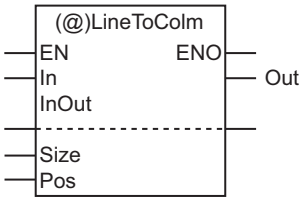
Для вывода каждого бита битовой строки в соответствующий бит в указанной позиции каждого элемента массива используйте команду *LineToColm* на стр. 2-476.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *Size* = *0*, то все биты в *Out* переходят в состояние ЛОЖЬ.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Size* находится за пределами допустимого диапазона.
  b) Значение *Pos* находится за пределами допустимого диапазона.
  c) Значение *Size* приводит к выходу за область массива In[].

LineToColm

Команда LineToColm извлекает биты из битовой строки и выводит каждый из них в соответствующий бит в указанной позиции каждого элемента массива.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LineToColm	Преобразование строки в столбец	FUN		LineToColm(In, InOut, Size, Pos);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	*1
Size	Количество элементов в результате		Количество элементов в результате	От 0 до количества битов в *In*		1
Pos	Положение бита для преобразования		Позиция бита, принимающего результат	От 0 до количества битов в InOut[] - 1		0
InOut[] (массив)	Массив результатов преобразования	Вход-выход	Результат преобразования	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		ОК	ОК	ОК	ОК															
Size						ОК														
Pos						ОК														
InOut[] (массив)		ОК	ОК	ОК	ОК															
Out	ОК																			

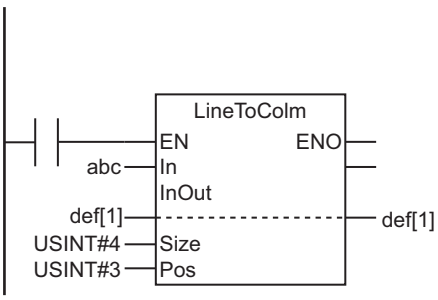
Функция

Команда LineToColm извлекает биты из битовой строки и выводит каждый из них в соответствующий бит в указанной позиции каждого элемента массива.

Сначала извлекаются *Size* младших битов из *In* (данные для преобразования), после чего команда оперирует ими как отдельными битами.
Затем каждый извлеченный бит сохраняется в *Pos*-й бит соответствующего элемента InOut[], начиная с InOut[0]. Значение *Size* равно количеству элементов массива, в которые сохраняются извлеченные биты.
Во всех остальных битах, в которые не сохраняются извлеченные значения, остаются прежние значения.

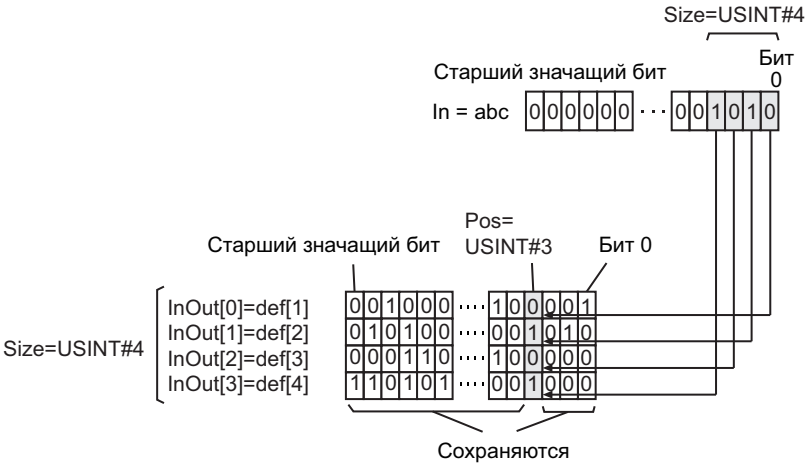
Ниже показан пример для случая, когда *Pos* = USINT#3, а *Size* = USINT#4.

LD



ST

```
LineToColm(abc, def[1], USINT#4, USINT#3);
```



Дополнительная информация

Для извлечения значений битов в указанной позиции каждого элемента массива и вывода их в виде битовой строки используйте команду *ColmToLine_*** на стр. 2-474.

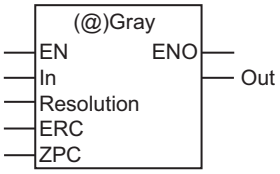
Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *Size* = *0*, то значения в InOut[] не изменятся.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.

- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое InOut[] не изменится.
  a) Значение *Size* находится за пределами допустимого диапазона.
  b) Значение *Pos* находится за пределами допустимого диапазона.
  c) Значение *Size* приводит к выходу за область массива InOut[].

Gray

Команда Gray преобразует код Грея в значение угла.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Gray	Преобразование кода Грея	FUN		Out:=Gray(In, Resolution, ERC, ZPC);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Код Грея для преобразования	Зависит от типа данных.	---	0
Resolution	Разрешение		Разрешение	_R256, _R1B..._R15B, _R360, _R720 или _R1024		_R256
ERC	Компенсация избытка энкодера		Компенсация избытка энкодера	От 0 до *Resolution*		0
ZPC	Коррекция нулевой точки		Коррекция нулевой точки			
Out	Результат преобразования	Выход	Результат преобразования	*1	°	---

*1. 0...3,59999999999999e+2

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In			OK																	
Resolution	Сведения о перечислителях перечислимого типа _eGRY_RESOLUTION см. в разделе *Функция* на стр. 2-479.																			
ERC							OK													
ZPC							OK													
Out															OK					

Функция

Команда Gray преобразует код Грея, содержащийся в *In* (выходное значение от углового энкодера), в значение угла. Переменная *Out* содержит результат преобразования в градусах.

Для параметра *Resolution* используется перечислимый тип данных _eGRY_RESOLUTION. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_R256	256
_R1B	1-разр. (2)
_R2B	2-разр. (4)
_R3B	3-разр. (8)
_R4B	4-разр. (16)
_R5B	5-разр. (32)
_R6B	6-разр. (64)
_R7B	7-разр. (128)
_R8B	8-разр. (256)
_R9B	9-разр. (512)
_R10B	10-разр. (1024)
_R11B	11-разр. (2048)
_R12B	12-разр. (4096)
_R13B	13-разр. (8192)
_R14B	14-разр. (16384)
_R15B	15-разр. (32768)
_R360	360
_R720	720
_R1024	1024

Код Грея

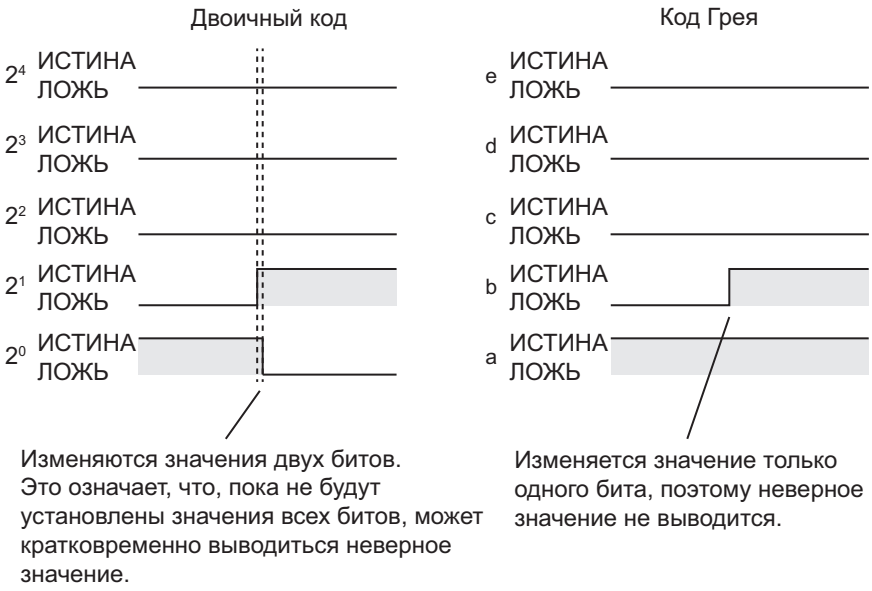
Код Грея представляет собой отраженный двоичный код.
Два последовательных значения, такие как 0 и 1 или 1 и 2, различаются только в одном бите.
Коды Грея используются для вывода данных в абсолютных энкодерах.

В таблице ниже представлены 4-разрядный двоичный код и код Грея.

Десятичное число	Двоичный код				Код Грея			
	2^3	2^2	2^1	2^0	d	c	b	a
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Использование кода Грея позволяет предотвратить кратковременный вывод неверных значений, поскольку при увеличении или уменьшении выходного значения энкодера на 1 в коде Грея изменяется только один бит.
На следующем рисунке показано, чем отличаются выходные значения энкодера при использовании кода Грея и двоичного кода.

Различия при изменении выходного значения с 1 на 2



Изменяются значения двух битов. Это означает, что, пока не будут установлены значения всех битов, может кратковременно выводиться неверное значение.

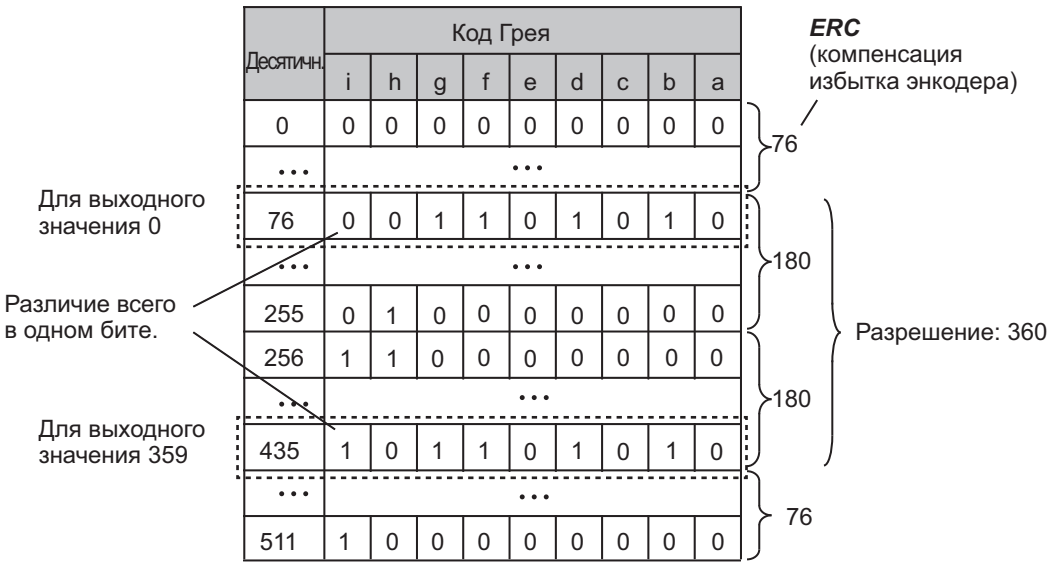
Изменяется значение только одного бита, поэтому неверное значение не выводится.

ERC: компенсация избытка энкодера

С помощью переменной *ERC* можно задать используемый диапазон значений в коде Грея, когда разрешение энкодера не выражается степенью числа 2. Диапазон указывается с таким расчетом, чтобы максимальное и минимальное выходные значения энкодера различались только в одном бите.

Пусть, например, используется абсолютный энкодер с разрешающей способностью 360. Будем использовать 9-разрядный код Грея (9 битов). С помощью 9 битов можно закодировать числа в диапазоне от 0 до 511, т. е. всего 512 чисел. Нас интересуют только 360 чисел, половину которых мы возьмем из левой половины имеющегося диапазона, а вторую половину из правой половины. Определим левое и правое крайние значения: 255 - 180 + 1 = 76; 256 + 180 - 1 = 435. Следовательно, используются значения от 76 до 435 в коде Грея. Для выходного значения 0 выдается значение в коде Грея 001101010 (76 в десятичной записи), а для выходного значения 359 выдается значение в коде Грея 101101010 (435 в десятичной записи). Эти значения различаются только в одном бите.

В данном случае значение параметра *ERC* (компенсация избытка энкодера) равно 76.



ZPC: коррекция нулевой точки

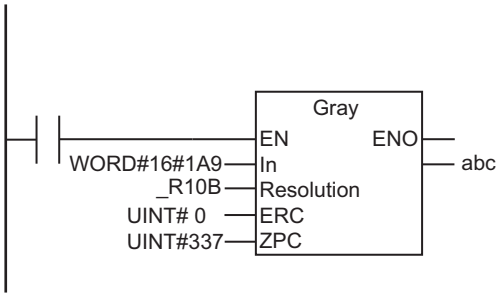
С помощью параметра *ZPC* можно задать смещение нулевого положения углового энкодера. Например, чтобы сместить нулевое положение углового энкодера с разрешением 256 на 90 градусов, значение *ZPC* должно быть равно 256 × (90/360) = 64.

Пример записи

Ниже представлен пример, в котором *In* = WORD#16#1A9, *Resolution* = _R10B, *ERC* = UINT#0, а *ZPC* = UINT#337.

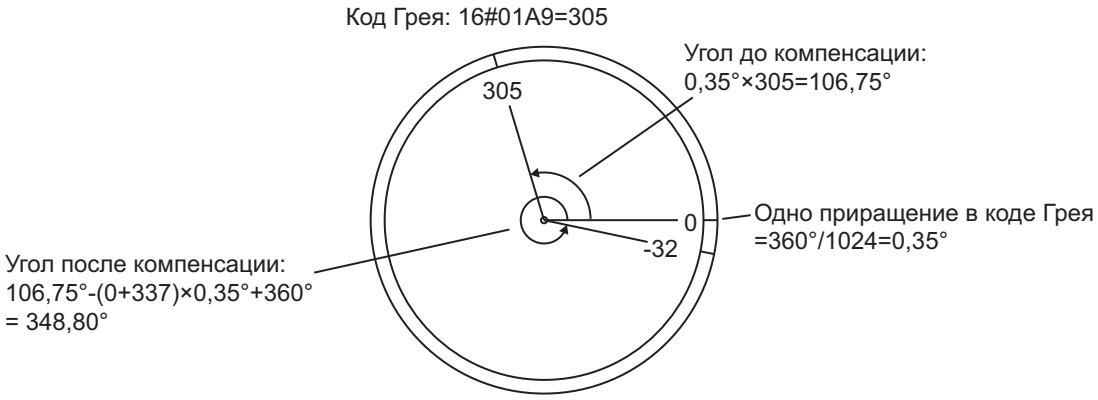
Так как разрешение составляет 10 разрядов (битов), одному приращению значения в коде Грея соответствует угол 360°/1024 = 0,35°.
Значению 16#01A9 в коде Грея соответствует десятичное значение 305. Следовательно, угол до компенсации равен 0,35° x 305 или 106,75°.
ERC = 0, а *ZPC* = 377. Значение угла после компенсации вычисляется следующим образом: 106,75° - (0 + 337) × 0,35° = -11,20°.
Значение *Out* должно быть больше или равно 0, поэтому значение вычисляется следующим образом: -11,20° + 360° = 348,80°. Таким образом, *Out* = LREAL#348.8.

LD



ST

```
abc:=Gray(WORD#16#1A9, _R10B, UINT#0, UINT#337);
```



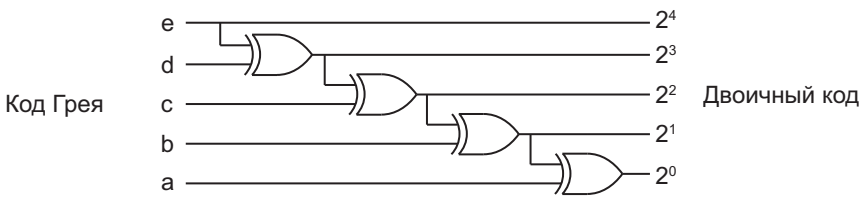
Дополнительная информация

При выборе значений параметров *Resolution* и *ERC* см. пользовательскую документацию по используемому угловому энкодеру.

Преобразование кода Грея в двоичный код

Для преобразования значения из кода Грея в двоичный код можно использовать показанную ниже схему. Символы логических элементов на рисунке представляют логическое исключающее ИЛИ.

Схема преобразования для пяти разрядов (битов)



Меры предосторожности для обеспечения надлежащей эксплуатации

В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Значение *Resolution* находится за пределами допустимого диапазона.
- Значение параметра *ERC* превышает разрешение, указанное в параметре *Resolution*.
- Значение параметра *ZPC* превышает разрешение, указанное в параметре *Resolution*.
- При преобразовании в битовую строку значение *In* меньше, чем значение *ERC*.
- Значение битовой строки после применения поправочного значения *ERC* превышает разрешение, указанное в параметре *Resolution*.

UTF8ToSJIS

Команда UTF8ToSJIS преобразует текстовую строку в кодировке UTF-8 в массив байтов (BYTE) с кодами символов в кодировке SJIS.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
UTF8ToSJIS	Преобразование кодировки UTF-8 в SJIS	FUN	(@)UTF8ToSJIS EN ENO In Out SJISCode	Out:=UTF8ToSJIS(In, SJISCode);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Текстовая строка для преобразования	Вход	Текстовая строка для преобразования	Зависит от типа данных.	---	"
SJISCode[] (массив)	Массив кодов SJIS	Вход-выход	Массив кодов символов SJIS	Зависит от типа данных.	---	---
Out	Количество преобразованных элементов	Выход	Количество элементов, сохраненных в массив SJISCode[]	0...1985	---	---

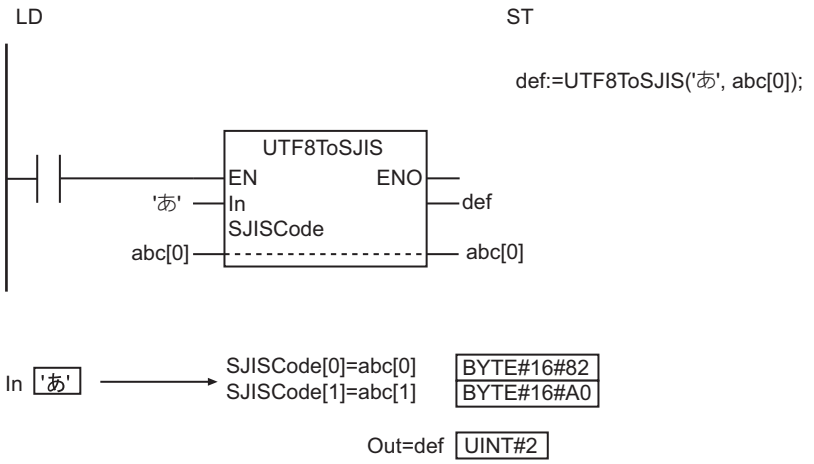
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
SJISCode[] (массив)		OK																		
Out							OK													

Функция

Команда UTF8ToSJIS преобразует текстовую строку в кодировке UTF-8 на входе *In* в массив SJISCode[], представляющий собой массив байтов (BYTE), содержащих соответствующие коды символов в кодировке SJIS. Преобразуемые данные разделяются на байты, каждый из которых по порядку сохраняется в соответствующий элемент массива SJISCode[], начиная с элемента SJISCode[0].

Количество элементов массива SJISCode[], в которых хранятся преобразованные данные, присваивается переменной *Out* (количество преобразованных элементов).

В показанном ниже примере программы *In* = «あ».

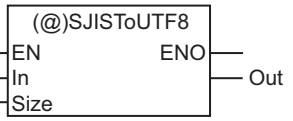


Меры предосторожности для обеспечения надлежащей эксплуатации

- Нулевые символы (NULL) в конце строки *In* не преобразуются. Они также не учитываются в количестве преобразованных элементов.
- Если текстовая строка *In* содержит только нулевые символы (NULL), то значение *Out* будет равно 0, а содержимое SJISCode[] не изменится.
- Содержимое элементов массива SJISCode[], расположенных после *Out* элементов, не изменяется. Например, если количество преобразованных элементов равно 5, то элемент SJISCode[5] и следующие за ним элементы не изменяются.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значение *Out* и содержимое SJISCode[] не изменятся.
  a) Количество преобразованных элементов выходит за диапазон значений выходного параметра для SJISCode[].
  b) Строка *In* включает символы, которые не могут быть преобразованы.

SJISToUTF8

Команда SJISToUTF8 преобразует массив байтов, содержащих коды символов в кодировке SJIS, в текстовую строку в кодировке UTF-8.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SJISToUTF8	Преобразование кодировки SJIS в UTF-8	FUN		Out:=SJISToUTF8(In, Size);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив кодов SJIS для преобразования	Вход	Массив кодов в кодировке SJIS для преобразования*1	Зависит от типа данных.	---	*2
Size	Количество элементов в массиве кодов SJIS		Количество элементов в массиве In[] для преобразования			---
Out	Результирующая текстовая строка	Выход	Текстовая строка в кодировке UTF-8 после преобразования	Зависит от типа данных.	---	---

*1. Максимальное число элементов составляет 1986, включая нулевой символ (NULL) (BYTE#16#00). Без нулевого символа (NULL) максимальное число элементов составляет 1985.

*2. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

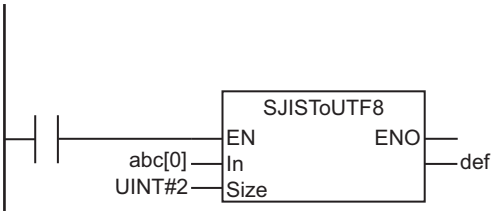
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)		ОК																		
Size							ОК													
Out																				ОК

Функция

Команда SJISToUTF8 преобразует элементы массива In[] типа BYTE (массив SJIS для преобразования) в текстовую строку в кодировке UTF-8.
Преобразуются *Size* элементов массива In[], начиная с элемента In[0]. Однако если какой-либо из элементов содержит нулевой символ (NULL) (BYTE#16#0), преобразование завершается в этом месте.
Преобразованная текстовая строка сохраняется в *Out* (результирующая текстовая строка). В конец строки *Out* добавляется нулевой символ (NULL).

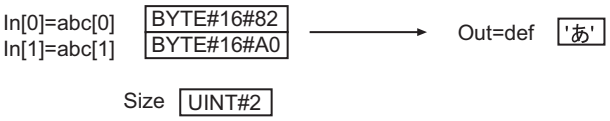
В показанном ниже примере программы In[0] = BYTE#16#82, In[1] = BYTE#16#A0, а *Size* = UINT#2.

LD



ST

```
def:=SJISToUTF8(abc[0], UINT#2);
```



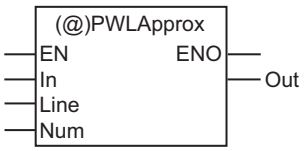
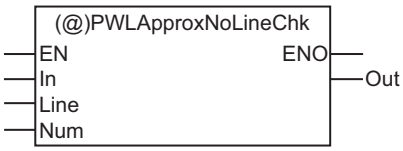
Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *Size* = 0, то *Out* представляет собой текстовую строку, содержащую только нулевые символы (NULL).
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Size* превышает количество элементов массива In[].
  b) Элементы массива In[] включают символы, которые не могут быть преобразованы.

PWLApprox и PWLApproxNoLineChk

Команды PWLApprox и PWLApproxNoLineChk аппроксимируют целочисленные или вещественные координаты ломаной линии.

PWLApprox : Проверяет достоверность данных ломаной линии.
PWLApproxNoLineChk : Не проверяет достоверность данных ломаной линии.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
PWLApprox	Аппроксимация ломаной линии с проверкой данных ломаной линии	FUN		Out:=PWLApprox(In, Line, Num);
PWLApproxNoLineChk	Аппроксимация ломаной линии без проверки данных ломаной линии	FUN		Out:=PWLApproxNoLineChk(In, Line, Num);

Информация о версии

Для использования команды PWLApproxNoLineChk требуются модуль ЦПУ с версией модуля 1.03 или более поздней и Sysmac Studio версии 1.04 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	*1
Line[] (массив)	Массив данных ломаной линии		Массив данных ломаной линии			
Num	Количество данных ломаной линии		Количество данных ломаной линии			1
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

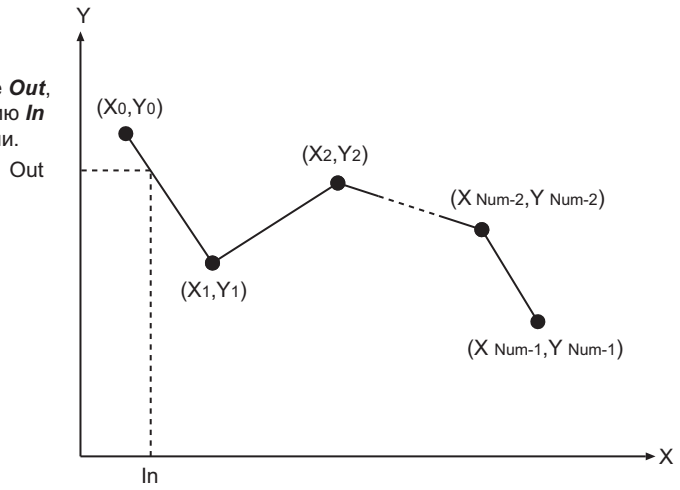
*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
Line[] (массив)	Должен представлять собой массив с элементами того же типа данных, что и значение *In*.																			
Num							ОК													
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					

Функция

Команды PWLApprox и PWLApproxNoLineChk выполняют аппроксимацию для значения *In* (данные для преобразования). Аппроксимация выполняется на основе данных ломаной линии в массиве Line[] (массив данных ломаной линии). Данные ломаной линии содержат *Num* умножить на 2 элементов и начинаются с элемента Line[0,0].
Как видно из примера ниже, в переменную *Out* записывается координата Y точки на ломаной линии, которая соответствует координате X, введенной в переменную *In*.

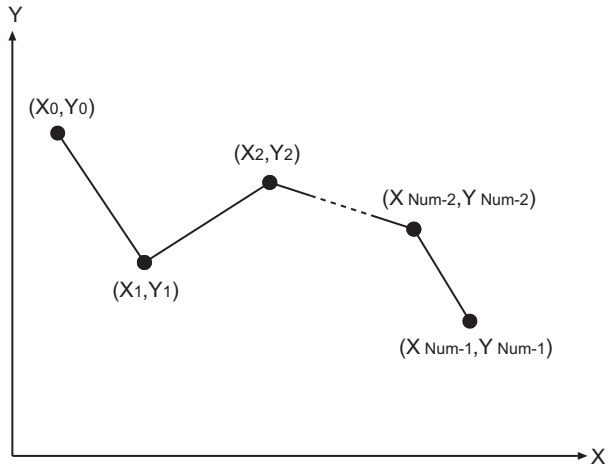
Команда найдет значение ***Out***, соответствующее значению ***In*** на графике ломаной линии.



Элементы массива данных ломаной линии Line[] и количество данных ломаной линии *Num*

Массив Line[] должен быть двумерным или трехмерным массивом. Для первого измерения количество элементов следует установить равным 2. Элементы массива Line[] должны содержать значения координат точек ломаной линии: $(X_0,Y_0),(X_1,Y_1)$ и т. д. (см. пример на рисунке ниже).

Количество данных ломаной линии *Num* равно половине количества элементов массива Line[], который используется для приближенного расчета (аппроксимации) координат ломаной линии.



Использование двумерного массива для ***Line[]***

Line[0,0]	X_0
Line[0,1]	Y_0
Line[1,0]	X_1
Line[1,1]	Y_1
Line[2,0]	X_2
Line[2,1]	Y_2
:	:
Line[Num-1,0]	X Num-1
Line[Num-1,1]	Y Num-1

Использование трехмерного массива для ***Line[]***

Line[0,0,0]	X_0
Line[0,0,1]	Y_0
Line[0,1,0]	X_1
Line[0,1,1]	Y_1
Line[0,2,0]	X_2
Line[0,2,1]	Y_2
:	:
Line[0, Num-1,0]	X Num-1
Line[0, Num-1,1]	Y Num-1

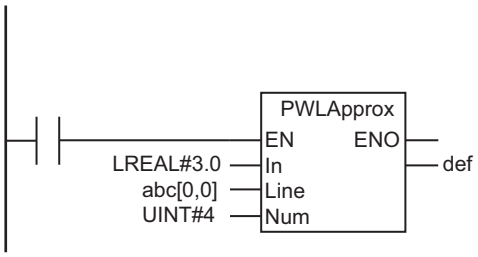
Пример записи

Ниже представлен пример аппроксимации координаты Y для значения координаты X в *In* = LREAL 3.0 на основе массива данных ломаной линии abc[] с четырьмя элементами. В данном примере *Num* = UINT#4, а элементы массива abc[] содержат следующие значения:

- abc[0.0] = X_0 = LREAL#1.0, abc[0,1] = Y_0 = LREAL#5.0,
- abc[1.0] = X_1 = LREAL#2.0, abc[1,1] = Y_1 = LREAL#6.0,
- abc[2.0] = X_2 = LREAL#4.0, abc[2,1] = Y_2 = LREAL#2.0,
- abc[3.0] = X_3 = LREAL#5.0, abc[3,1] = Y_3 = LREAL#3.0

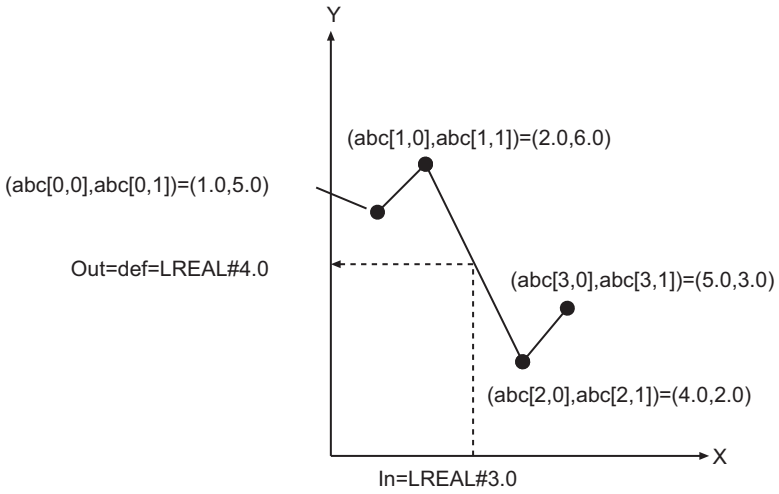
Результатом вычислений является значением LREAL#4.0, которое присваивается переменной *Out*.

LD



ST

```
def:=PWLApprox(LREAL#3.0, abc[0,0], UINT#4);
```



Различия между командами PWLApprox и PWLApproxNoLineChk

Команды PWLApprox и PWLApproxNoLineChk отличаются друг от друга тем, что одна из них проверяет действительность значений *In* и Line[], а вторая — нет. Соответственно, они отличаются и временем выполнения. Характеристики обеих команд приведены в следующей таблице.

Команда	Проверка	Действия при недействительных значениях	Время выполнения
PWLApprox	• Команда проверяет, что элементы массива Line[] упорядочены в порядке возрастания значений координаты X. • Если *In* и Line[] являются целочисленными, команда проверяет, не содержат ли *In* или какой-либо из элементов массива Line[] нечисловое значение либо положительную или отрицательную бесконечность.	• Возникает ошибка. • Выход *ENO* находится в состоянии ЛОЖЬ. • Значение *Out* не изменяется.	Долго
PWLApproxNoLineChk	Никаких проверок не производится.	• Ошибки не произойдет. • Выход *ENO* будет находиться в состоянии ИСТИНА. • В *Out* может быть не выведено верное значение.	Быстро

Команды PWLApproxNoLineChk и PWLLineChk

Так как команда PWLApproxNoLineChk не проверяет действительность значений *In* и Line[], она выполняется быстрее. Так что, если входные переменные заведомо действительны, лучше использовать команду PWLApproxNoLineChk, а не команду PWLApprox.

Команда *PWLLineChk* на стр. 2-494 проверяет содержимое массива Line[] на предмет того, что значения координаты X располагаются в порядке возрастания. Время обработки можно уменьшить, используя команду PWLApproxNoLineChk в обычном случае и применяя команду PWLLineChk лишь тогда, когда нужно проверить, что данные в массиве Line[] упорядочены в порядке возрастания значений координаты X.

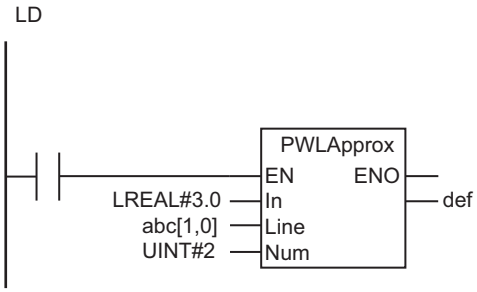
Дополнительная информация

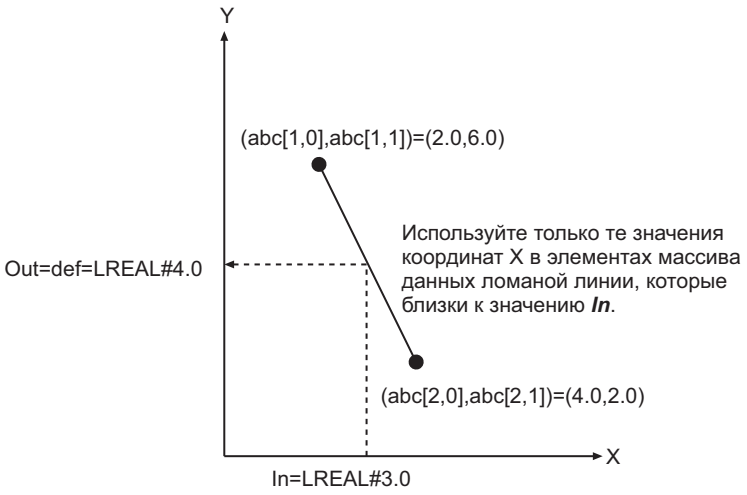
Время обработки также можно сократить, ограничив количество элементов в массиве данных ломаной линии, используемых для аппроксимации.

В предыдущем примере для того же значения *In* (LREAL#3.0) время обработки будет меньше, если аппроксимация будет выполняться по указанным ниже четырем элементам, содержащим значения координаты X, близкие к 3,0.

(abc[1,0], abc[1,1]) = (2.0, 6.0)
(abc[2,0], abc[2,1]) = (4.0, 2.0)

В данном случае *Num* = UINT#2, а в параметр Line[] передается элемент abc[1,0] массива abc[]. Результат преобразования в переменной *Out* по-прежнему будет равен LREAL#4.0.





Меры предосторожности для обеспечения надлежащей эксплуатации

- Если значение *In* меньше, чем значение Line [0,0] (т. е. значение X_1), то переменной *Out* будет присвоено значение Line[0,1] (т. е. значение Y_1).
- Если значение *In* больше, чем значение Line[*Num*-1,0] (т. е. значение X_{Num}), то переменной *Out* будет присвоено следующее значение:

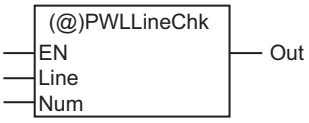
  Line[*Num*-1,1] (т. е. значение Y_{Num})

- Массив Line[] должен быть двумерным или трехмерным массивом. Для первого измерения количество элементов следует установить равным 2.
- Если значение *Num* равно 0, то значение *Out* также равно 0.

- В указанных ниже случаях при использовании команды PWLApprox произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится. При использовании команды PWLApproxNoLineChk в этих случаях ошибки не произойдет.
  a) Значения координаты X в данных ломаной линии не располагаются в порядке возрастания; условие $X_1 < X_2 < ... < X_{Num}$ не соблюдается.
  b) *In* и Line[] являются вещественными (REAL), при этом среди их значений имеются нечисловое значение либо положительная или отрицательная бесконечность.
- В указанных ниже случаях при использовании команды PWLApprox или PWLApproxNoLineChk произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Num* приводит к выходу за область массива Line[].
  b) Значение *In* находится за пределами области координат X, указанных в массиве данных ломаной линии Line[].

PWLLineChk

Команда PWLLineChk проверяет, упорядочены ли данные ломаной линии, используемые для команды PWLApproxNoLineCheck, в порядке возрастания значений координаты X.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
PWLLineChk	Проверка данных ломаной линии	FUN		Out:=PWLLineChk(Line, Num);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.03 или более поздней и Sysmac Studio версии 1.04 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Line[] (массив)	Массив данных ломаной линии	Вход	Массив данных ломаной линии	Зависит от типа данных.	---	*1
Num	Количество данных ломаной линии		Количество данных ломаной линии			1
Out	Результат	Выход	Результат	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Line[] (массив)						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
Num							ОК													
Out	ОК																			

Функция

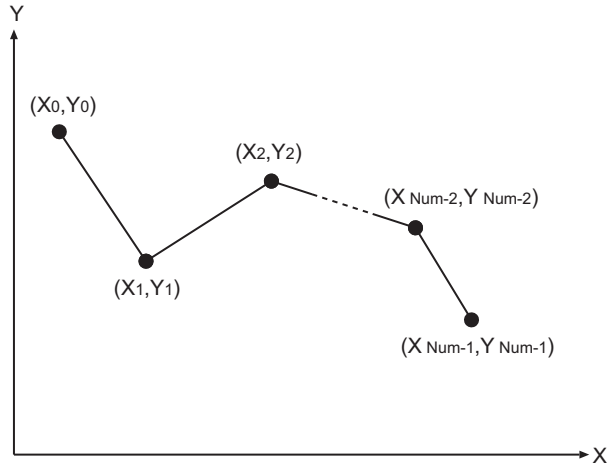
С помощью команды PWLLineChk можно проверить, упорядочены ли в порядке возрастания значения координаты X в массиве данных ломаной линии Line[], который используется для команды PWLApproxNoLineChk (Аппроксимация ломаной линии без проверки данных ломаной линии).

Если значения координаты X расположены в порядке возрастания, то на выход *Out* (результат) подается значение ИСТИНА. Если же нет, то выход *Out* находится в состоянии ЛОЖЬ.

Элементы массива данных ломаной линии Line[] и количество данных ломаной линии *Num*

Массив Line[] должен быть двумерным или трехмерным массивом. Для первого измерения количество элементов следует установить равным 2. Элементы массива Line[] должны содержать значения координат точек ломаной линии: $(X_0,Y_0),(X_1,Y_1)$ и т. д. (см. пример на рисунке ниже).

Количество данных ломаной линии *Num* равно половине количества элементов массива Line[], который используется для приближенного расчета (аппроксимации) координат ломаной линии.



Использование двумерного массива для ***Line[]***

Line[0,0]	X_0
Line[0,1]	Y_0
Line[1,0]	X_1
Line[1,1]	Y_1
Line[2,0]	X_2
Line[2,1]	Y_2
:	:
Line[Num-1,0]	X Num-1
Line[Num-1,1]	Y Num-1

Использование трехмерного массива для ***Line[]***

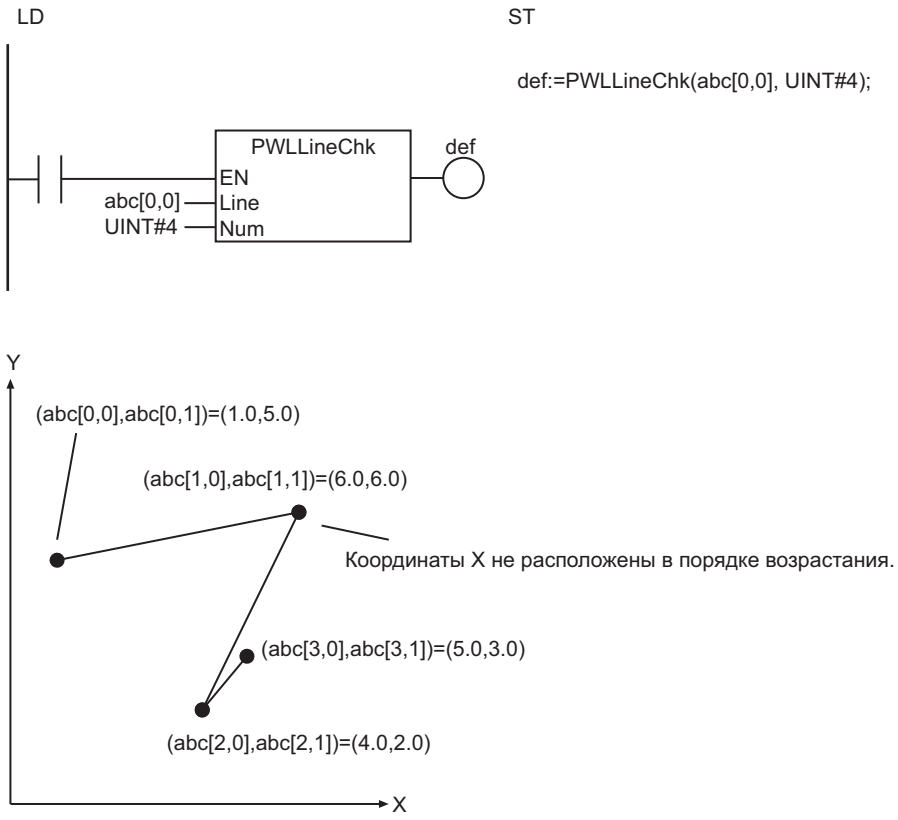
Line[0,0,0]	X_0
Line[0,0,1]	Y_0
Line[0,1,0]	X_1
Line[0,1,1]	Y_1
Line[0,2,0]	X_2
Line[0,2,1]	Y_2
:	:
Line[0, Num-1,0]	X Num-1
Line[0, Num-1,1]	Y Num-1

Пример записи

В представленном ниже примере проверяется, упорядочены ли четыре элемента массива abc[] (массив данных ломаной линии) в порядке возрастания значений координаты X. В данном примере *Num* = UINT#4, а элементы массива abc[] содержат следующие значения:

- abc[0.0] = X_0 = LREAL#1.0, abc[0,1] = Y_0 = LREAL#5.0,
- abc[1.0] = X_1 = LREAL#6.0, abc[1,1] = Y_1 = LREAL#6.0,
- abc[2.0] = X_2 = LREAL#4.0, abc[2,1] = Y_2 = LREAL#2.0,
- abc[3.0] = X_3 = LREAL#5.0, abc[3,1] = Y_3 = LREAL#3.0

Значения координаты X не упорядочены в порядке возрастания, поэтому на выход *Out* подается ЛОЖЬ.



Дополнительная информация

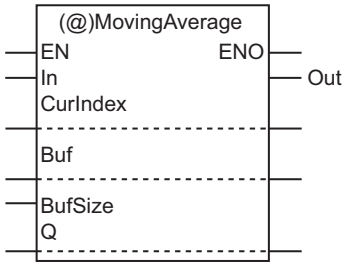
- Используйте данную команду совместно с командой PWLApproxNoLineChk. Сведения о команде PWLApproxNoLineChk см. в разделе *PWLApprox и PWLApproxNoLineChk* на стр. 2-488.
- Если данные ломаной линии требуется проверять при каждом выполнении аппроксимации ломаной линии, используйте команду PWLApprox. Сведения о команде PWLApprox см. в разделе *PWLApprox и PWLApproxNoLineChk* на стр. 2-488. Команда PWLApproxNoLineChk выполняется быстрее, чем команда PWLApprox.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Массив Line[] должен быть двумерным или трехмерным массивом. Для первого измерения количество элементов следует установить равным 2.
- В указанных ниже случаях произойдет ошибка. *Out* переходит в состояние ЛОЖЬ.
  a) Значение *Num* приводит к выходу за область массива Line[].
  b) Массив Line[] является вещественным (REAL), при этом его элементы содержат нечисловое значение либо положительную или отрицательную бесконечность.

MovingAverage

Команда MovingAverage вычисляет скользящее среднее значение.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MovingAverage	Скользящее среднее	FUN		Out:=MovingAverage(In, CurIndex, Buf, BufSize, Q);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Входное значение	Вход	Число, включаемое в вычисление среднего значения	Зависит от типа данных.	---	*1
BufSize	Максимальное количество значений		Максимальное количество элементов для вычисления среднего значения			1
CurIndex	Позиция хранения входного значения	Вход-выход	Позиция в Buf[] для хранения *In*	Зависит от типа данных.	---	---
Buf[] (массив)	Массив хранения входных значений		Массив для хранения значений *In*			
Q	Флаг завершения вычисления		ИСТИНА: количество значений, сохраненных в Buf[], достигло или превысило *BufSize*. ЛОЖЬ: количество значений, сохраненных в Buf[], еще не достигло *BufSize*.			
Out	Результат вычисления	Выход	Результат вычисления	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

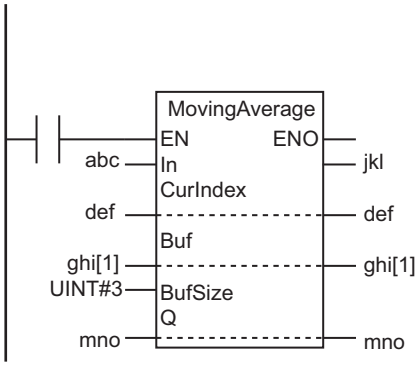
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
BufSize							ОК													
CurIndex							ОК													
Buf[] (массив)	Должен представлять собой массив с элементами того же типа данных, что и значение *In*.																			
Q	ОК																			
Out						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					

Функция

Команда MovingAverage сохраняет значение *In* в массив Buf[] (массив хранения входных значений) при каждом выполнении. Затем она вычисляет среднее значение введенных значений и сохраняет результат в *Out* (результат вычислений). В параметре *BufSize* указывается максимальное количество элементов, которые включаются в расчет среднего значения.

Далее будет рассмотрен порядок вычислений на примере, в котором *BufSize* = UINT#3. Вид команды для этого случая показан ниже.

LD

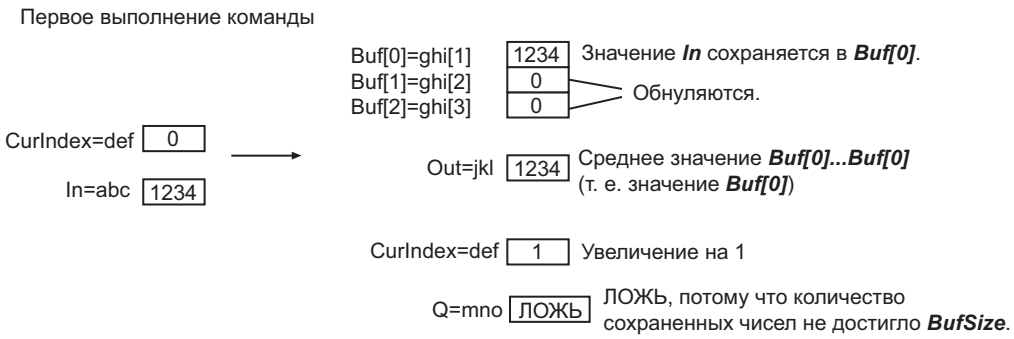


ST

```
jkl:=MovingAverage(abc, def, ghi[1], UINT#3, mno);
```

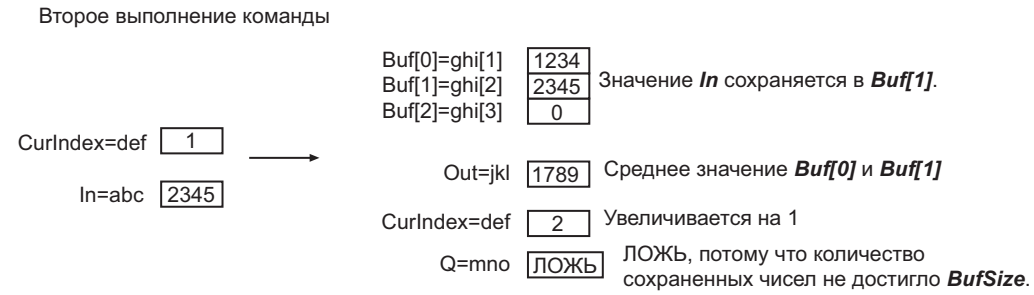
Число вводится в первый раз

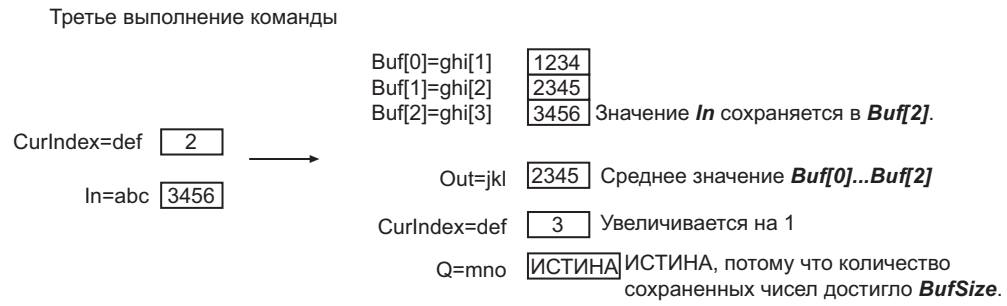
В переменную *CurIndex* (позиция хранения входного значения) записывается значение 0, и выполняется команда.
В элементы с Buf[0] по Buf[*BufSize*-1] массива Buf[] (массив хранения входных значений) записываются нули, а затем в элемент Buf[0] сохраняется первое значение переменной *In* (входное значение).
Значение переменной *Q* (флаг завершения вычислений) меняется на ЛОЖЬ. Это означает, что число значений, сохраненных в массив Buf[], еще не достигло значения *BufSize*.
Пока флаг *Q* остается в состоянии ЛОЖЬ, среднее значение вычисляется по *CurIndex* + 1 числам, начиная с Buf[0]. Результат вычисления сохраняется в *Out*.
Наконец, значение *CurIndex* увеличивается на 1.



Ввод чисел до достижения *BufSize*

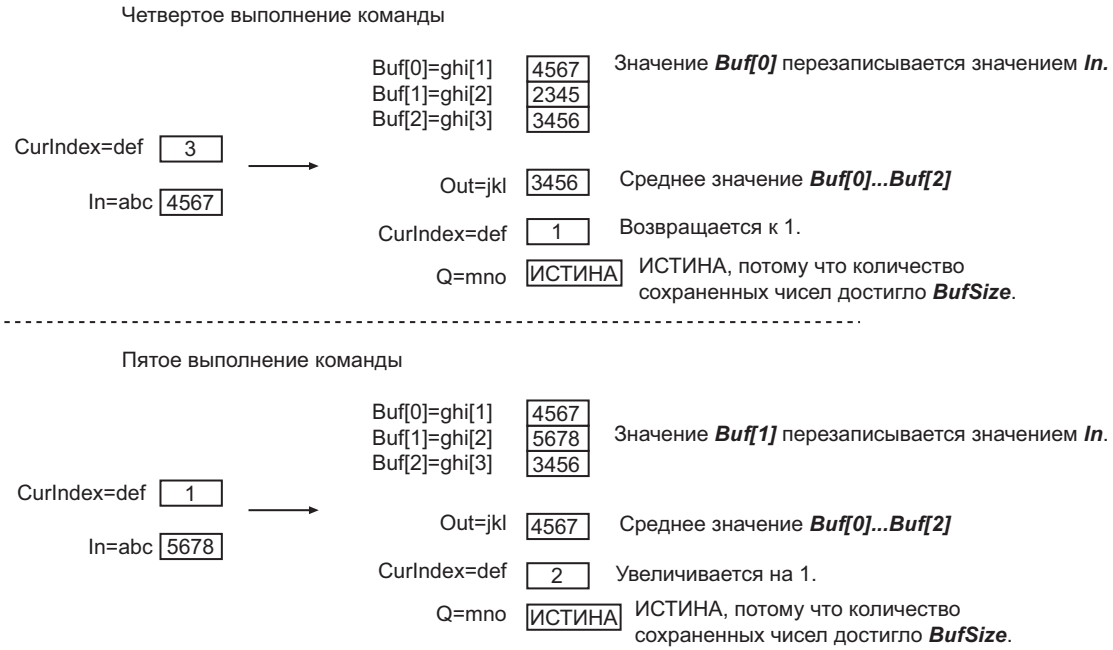
Каждый раз, когда выполняется команда, значение *In* сохраняется в элемент Buf[*CurIndex*] (начиная с элемента Buf[0]). Команда вычисляет среднее значение для *CurIndex* + 1 уже введенных значений и сохраняет результат в *Out*.
После того как команда оказывается выполнена *BufSize* раз, значение *Q* меняется на ИСТИНА.





Ввод чисел после достижения *BufSize*

Каждый раз, когда выполняется команда, в соответствующий элемент массива (с Buf[0] по Buf[*BufSize*-1]) вместо прежнего значения записывается новое значение *In*. Массив, таким образом, циклически перезаписывается. Вычисляется среднее значение по значениям элементов Buf[0]...Buf[*BufSize*-1], которое сохраняется в *Out*.
После достижения значения *BufSize* значение *CurIndex* возвращается к 1, а затем снова начинает увеличиваться. *Q* остается в состоянии ИСТИНА.

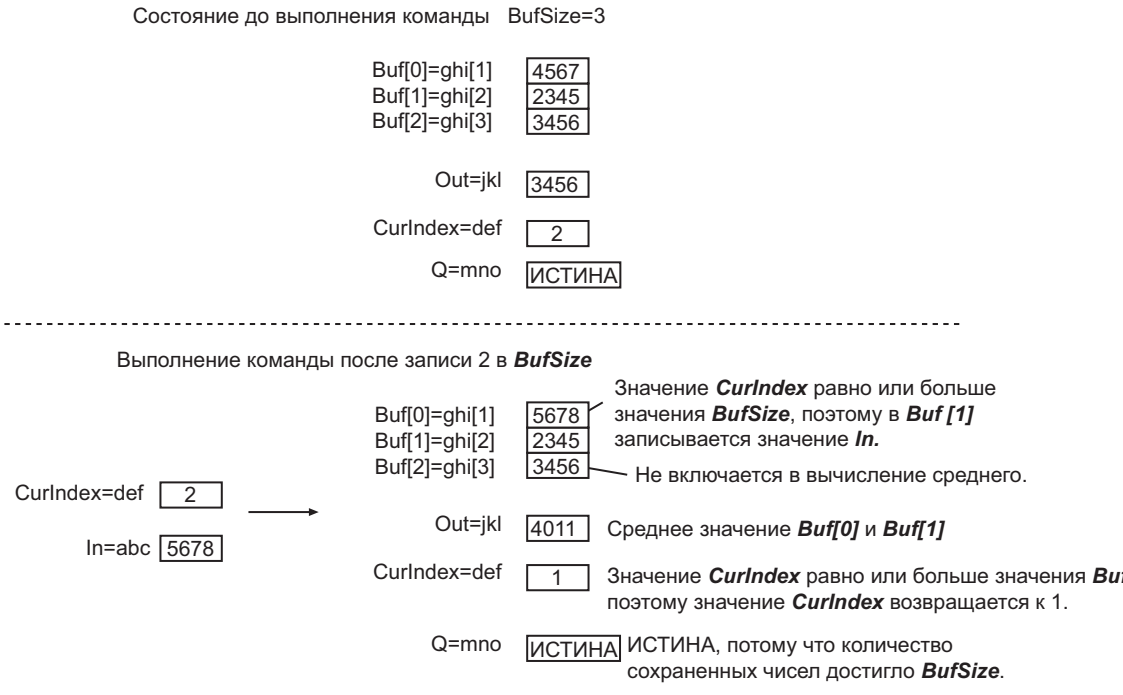


Инициализация сохраненных значений

Если в переменную *CurIndex* записать 0 до выполнения команды, все элементы с Buf[0] по Buf[*BufSize*-1] будут один раз обнулены, после чего в Buf[0] будет записано текущее значение *In*. Значение *CurIndex* станет равно 1, а значение *Q* поменяется на ЛОЖЬ.

Изменение значения *BufSize*

Если изменить значение *BufSize* и после этого выполнить команду, операция будет выполнена с новым значением *BufSize* и текущим значением *CurIndex*.

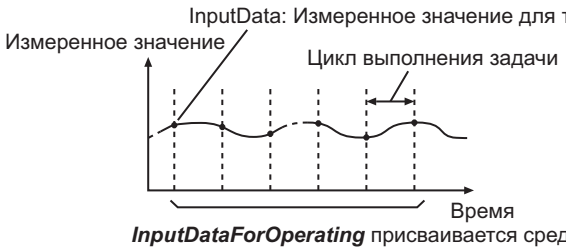


Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте одинаковый тип данных для переменной *In* и элементов Buf[]. При использовании разных типов данных произойдет ошибка сборки.
- Длина массива Buf[] должна быть не меньше значения *BufSize*.
- Даже если результат вычисления превысит диапазон допустимых значений *Out*, ошибки не произойдет. *Out* будет содержать недопустимое значение.
- Если значение *BufSize* равно 0, то значения *Out* и *CurIndex* также будут равны 0. Значение *Q* поменяется на ИСТИНА.
- Если вы изменяете значение *BufSize*, обязательно запишите 0 в *CurIndex* и инициализируйте сохраненные значения.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *BufSize* превышает длину массива Buf[].

Пример программы

В качестве примера рассмотрим фрагмент программы, который устраняет воздействие помех и других возмущений на входные аналоговые данные, получаемые, например, от датчика.
Вычисляется среднее значение по последним 25 значениям *InputData* (входные данные). Полученное среднее значение сохраняется в переменную *DataAve*, а та, в свою очередь, присваивается переменной *InputDataForOperating* для использования в качестве входного значения в следующей операции.
Значение *InputData* вводится в каждом цикле выполнения задачи при условии, что переменная *Trigger* (условие выполнения) = ИСТИНА.
До тех пор, пока не введены 25 значений *InputData* для вычисления среднего значения, переменной *InputDataForOperating* вместо среднего значения присваивается самое последнее значение переменной *InputData*.
Когда переменная *Trigger* переходит в состояние ИСТИНА, вычисленное среднее значение обнуляется и ввод значений *InputData* начинается с самого начала.



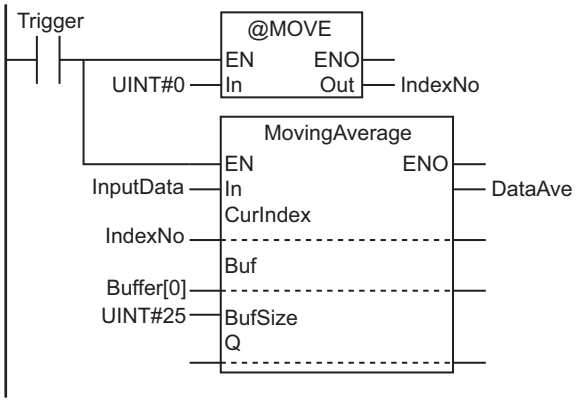
InputDataForOperating присваивается среднее значение по последним 25 значениям.

Программа на языке LD

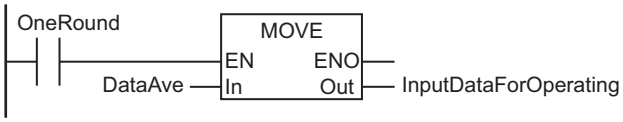
Переменная	Тип данных	Начальное значение	Комментарий
Trigger	BOOL	ЛОЖЬ	Условие выполнения
InputData	INT	10	Входное значение
Buffer	ARRAY[0..24] OF INT	[25(0)]	Массив хранения входных значений
DataAve	INT	0	Среднее значение
OneRound	BOOL	ЛОЖЬ	Флаг, указывающий на ввод 25 значений
IndexNo	UINT	0	Позиция хранения входного значения

Переменная	Тип данных	Начальное значение	Комментарий
InputDataForOperating	INT	0	Ввод значения для следующей операции

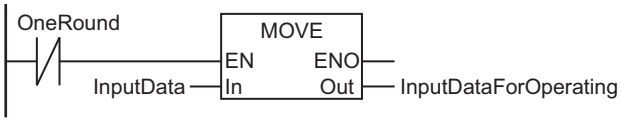
Когда состояние ***Trigger*** меняется на ИСТИНА, ***IndexNo*** присваивается 0.
Пока ***Trigger*** = ИСТИНА, в каждом цикле выполнения задачи вводится значение ***InputData*** и вычисляется среднее значение.



Когда имеется 25 или более входных значений для ***InputData***, ***DataAve*** присваивается ***InputDataForOperating***.



До тех пор пока не будет 25 или более входных значений для ***InputData***, ***InputData*** будет присваиваться ***InputDataForOperating***.



Программа на языке ST

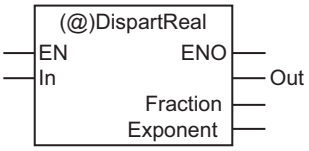
Переменная	Тип данных	Начальное значение	Комментарий
Trigger	BOOL	ЛОЖЬ	Условие выполнения
LastTrigger	BOOL	ЛОЖЬ	Значение *Trigger* в предыдущем цикле выполнения задачи
Operating	BOOL	ЛОЖЬ	Обработка
OperatingStart	BOOL	ЛОЖЬ	Обработка началась
Buffer	ARRAY[0..24] OF INT	[25(0)]	Массив хранения входных значений
InputData	INT	10	Входное значение
DataAve	INT	0	Среднее значение
OneRound	BOOL	ЛОЖЬ	Флаг, указывающий на ввод 25 значений
IndexNo	UINT	0	Позиция хранения входного значения
InputDataForOperating	INT	0	Ввод значения для следующей операции

```
// Определение перехода Trigger в состояние ИСТИНА.
IF ((Trigger=TRUE) AND (LastTrigger=FALSE)) THEN
    OperatingStart:=TRUE;
```

```
    Operating:=TRUE;
END_IF;
LastTrigger:=Trigger;

// Обнуление среднего значения.
IF (OperatingStart=TRUE) THEN
    IndexNo:=UINT#0;
    OperatingStart:=FALSE;
END_IF;

// Вычисление скользящего среднего.
IF (Operating=TRUE) THEN
    DataAve:=MovingAverage(
        In :=InputData,
        CurIndex:=IndexNo,
        Buf :=Buffer[0],
        BufSize :=UINT#25,
        Q :=OneRound);
    IF (OneRound=TRUE) THEN
        // Присвоение среднего значения по последним 25 значениям переменной InputD
ataForOperating.
        InputDataForOperating:=DataAve;
    ELSE
        // Присвоение самого последнего значения переменной InputDataForOperating.
        InputDataForOperating:=InputData;
    END_IF;
END_IF;

// Завершение вычисления среднего значения.
IF (Trigger=FALSE) THEN
    Operating:=FALSE;
END_IF;
```

DispartReal

Команда DispartReal разделяет вещественное число на мантиссу со знаком и показатель степени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DispartReal	Разделение на мантиссу и показатель степени	FUN		Out:=DispartReal(In, Fraction, Exponent);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Вещественное число	Вход	Вещественное число для разделения	Зависит от типа данных.	---	*1
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---
Fraction	Мантисса со знаком		Мантисса со знаком	*2		
Exponent	Показатель степени		Показатель степени	*3		

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

*2. Диапазоны допустимых значений зависят от типов данных *In* и *Fraction*. Дополнительные сведения см. в разделе *Допустимый диапазон для Fraction* на стр. 2-505.

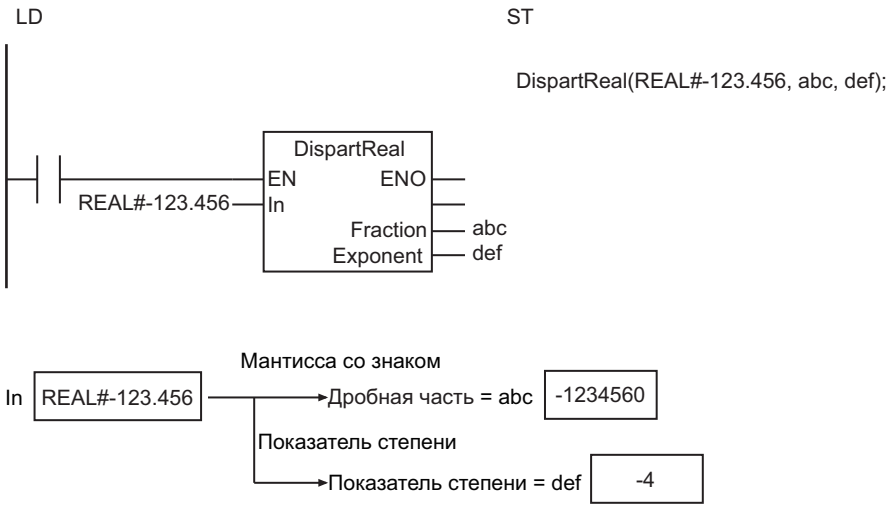
*3. Если *In* имеет тип данных REAL, то допускаются значения от -44 до 32, а если LREAL, то от -322 до 294.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In														ОК	ОК					
Out	ОК																			
Fraction	Если *In* — значение типа REAL, то должен использоваться тип DINT, а если LREAL, то LINT.																			
Exponent											ОК									

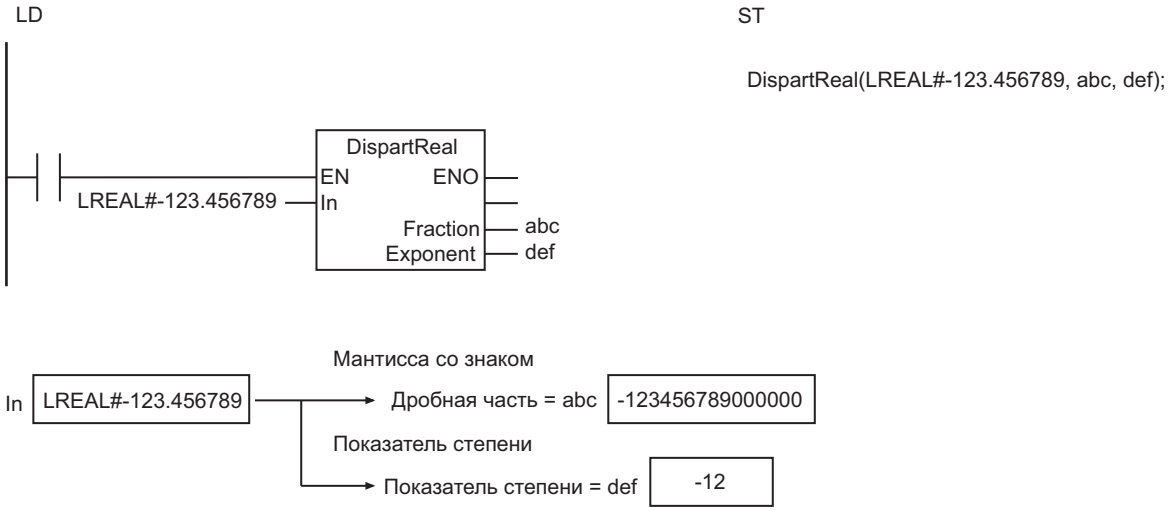
Функция

Команда DispartReal разделяет значение *In* (вещественное число) на мантиссу со знаком и показатель степени (*Exponent*).

Если *In* — значение типа REAL, то *Fraction* представляет собой 7-значное целое число. Если *In* — значение типа LREAL, то *Fraction* представляет собой 15-значное целое число.
В показанном ниже примере программы *In* = REAL#-123.456.



В показанном ниже примере программы *In* = LREAL#-123.456789.



Допустимый диапазон для *Fraction*

В следующей таблице приведены диапазоны допустимых значений для переменной *Fraction* в соответствии с типами данных *In* и *Fraction*.

Тип данных *In*	Тип данных *Fraction*	Диапазон допустимых значений *Fraction*
REAL	DINT	-9999999...9999999
LREAL	LINT	-999999999999999...999999999999999

Дополнительная информация

Для объединения мантиссы со знаком и показателя степени в вещественное число используйте команду *UniteReal* на стр. 2-507.

Меры предосторожности для обеспечения надлежащей эксплуатации

- В зависимости от значения переменной *In*, при преобразовании в целое число может возникнуть ошибка.
- Если количество действительных разрядов в *In* превосходит количество действительных разрядов во *Fraction*, значение округляется, чтобы соответствовать допустимому диапазону *Fraction*.
- Если в *In* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *In*	Тип данных *In*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значения *Fraction* и *Exponent* не изменятся.
  a) *In* содержит нечисловое значение или бесконечность.

Округление

В следующей таблице показано, как округляются значения.

Значение дробной части	Описание	Примеры
Меньше 0,5	Дробная часть отбрасывается.	1,49 → 1 -1,49 → -1
0,5	Если разряд единиц содержит четное число, дробная часть отбрасывается. Если это нечетное число, то значение округляется в большую сторону.	1,50 → 2 2,50 → 2 -1,50 → -2 -2,50 → -2
Больше 0,5	Дробная часть отбрасывается, число округляется в большую сторону.	1,51 → 2 -1,51 → -2

UniteReal

Команда UniteReal объединяет мантиссу со знаком и показатель степени и формирует из них вещественное число.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
UniteReal	Формирование вещественного числа из мантиссы и показателя степени	FUN	(@)UniteReal EN ENO Fraction Exponent Out	Out:=UniteReal(Fraction, Exponent);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Fraction	Мантисса со знаком	Вход	Мантисса со знаком	Зависит от типа данных.	---	*1
Exponent	Показатель степени		Показатель степени			0
Out	Вещественное число	Выход	Вещественное число	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

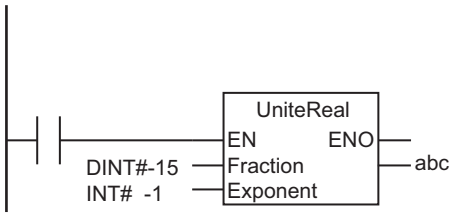
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Fraction												OK	OK							
Exponent											OK									
Out	Если *Fraction* — значение типа DINT, то должен использоваться тип REAL, а если LINT, то LREAL.																			

Функция

Команда UniteReal объединяет мантиссу со знаком *Fraction* и показатель степени *Exponent* и формирует из них вещественное число *Out*.

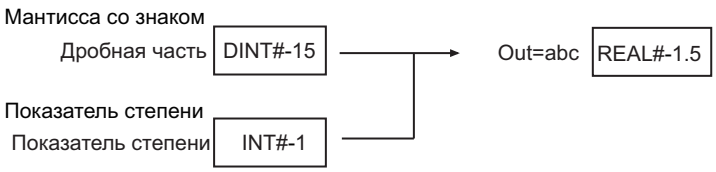
Ниже показан пример для случая, когда *Fraction* = DINT#-15, а *Exponent* = INT#-1.

LD



ST

```
abc:=UniteReal(DINT#15, INT#-1);
```



Дополнительная информация

Для разделения вещественного числа на мантиссу со знаком и показатель степени используйте команду *DispartReal* на стр. 2-504.

Меры предосторожности для обеспечения надлежащей эксплуатации

- В зависимости от значений переменных *Fraction* и *Exponent*, при преобразовании вещественного числа в целое число может возникнуть ошибка.
- Если результат объединения выходит за диапазон допустимых значений *Out* и *Exponent* содержит положительное число, значение *Out* является бесконечностью с тем же знаком, что и у значения *Fraction*. Если же *Exponent* содержит отрицательное число, значение *Out* будет равно 0.

NumToDecString и NumToHexString

NumToDecString : Преобразует целое число в текстовую строку фиксированной длины, содержащую число в десятичном виде.

NumToHexString : Преобразует целое число в текстовую строку фиксированной длины, содержащую число в шестнадцатеричном виде.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NumToDecString	Преобразование в текстовую строку фиксированной длины в десятичном виде	FUN	(@)NumToDecString EN ENO In Out L Fill	Out:=NumToDecString(In, L, Fill);
NumToHexString	Преобразование в текстовую строку фиксированной длины в шестнадцатеричном виде	FUN	(@)NumToHexString EN ENO In Out L Fill	Out:=NumToHexString(In, L, Fill);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Целое число	Вход	Целое число	Зависит от типа данных.	---	*1
L	Количество символов		Количество символов в *Out*	0...1985		1
Fill	Символ-заполнитель		Символ-заполнитель	_BLANK или _ZERO		_BLANK
Out	Текстовая строка	Выход	Текстовая строка	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						OK	OK	OK	OK	OK	OK	OK	OK							
L							OK													

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Fill	Сведения о перечислителях перечислимого типа _eFILL_CHR см. в разделе *Функция* на стр. 2-510.																			
Out																				OK

Функция

В обеих командах количество символов в текстовой строке *Out* определяется значением параметра *L* (количество символов).
В старшие позиции строки, не занятые цифрами, вставляется символ-заполнитель *Fill*.
Если количество символов в результате преобразования превышает *L*, в переменную *Out* сохраняются младшие *L* символов результата преобразования.
В конец строки *Out* добавляется нулевой символ (NULL). Нулевой символ в число символов не включается.

Для параметра *Fill* используется перечислимый тип данных _eFILL_CHR. Значения перечислителей приведены в таблице ниже.

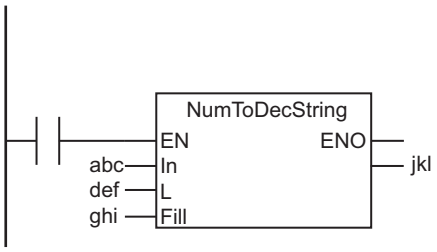
Перечислитель	Значение
_BLANK	« » (пробел)
_ZERO	«0»

NumToDecString

Команда NumToDecString преобразует целое число *In* в текстовую строку, содержащую символы десятичных цифр в кодировке UTF-8. Если *In* содержит отрицательное значение, то в начало текстовой строки добавляется знак минус (-).

Ниже приведены примеры для команды NumToDecString.

LD



ST

```
jkl:=NumToDecString(abc, def, ghi);
```

In = abc = INT#128, L = def = UINT#8, Fill = ghi = _BLANK
Out = jkl [ ][ ][ ][ ][1][2][8]

In = abc = INT#-128, L = def = UINT#8, Fill = ghi = _BLANK
Out = jkl [ ][ ][ ][ ][-][1][2][8]

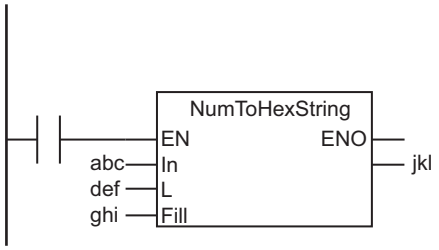
In = abc = INT#-128, L = def = UINT#8, Fill = ghi = _ZERO
Out = jkl [-][0][0][0][0][1][2][8]

NumToHexString

Команда NumToHexString преобразует целое число *In* в текстовую строку, содержащую символы шестнадцатеричных цифр в кодировке UTF-8. Если *In* содержит отрицательное значение, оно представляется в виде дополнения до двух (биты инвертируются, а затем добавляется 1).

Ниже приведены примеры для команды NumToHexString.

LD



ST

```
jkl:=NumToHexString(abc, def, ghi);
```

In = abc = INT#128, L = def = UINT#8, Fill = ghi = _BLANK
Out = jkl [ ][ ][ ][ ][ ][8][0]

In = abc = INT#128, L = def = UINT#8, Fill = ghi = _ZERO
Out = jkl [0][0][0][0][0][8][0]

In = abc = INT#-128, L = def = UINT#8, Fill = ghi = _BLANK
Out = jkl [F][F][F][F][F][8][0]

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *L* = 0, то *Out* представляет собой текстовую строку, содержащую только нулевые символы (NULL).
- Если количество символов в результате преобразования превышает значение *L*, в переменную *Out* сохраняются младшие *L* символов результата преобразования. Пример приведен ниже.

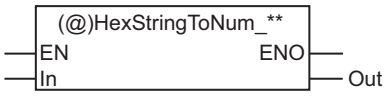
Команда	Значение *In*	Значение *L*	Значение *Out*
NumToDecString	128	2	28
NumToHexString			80

- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

a) Значение *L* находится за пределами допустимого диапазона.
b) Значение *Fill* находится за пределами допустимого диапазона.

HexStringToNum_**

Команда HexStringToNum_** преобразует текстовую строку, содержащую шестнадцатеричные цифры, в эквивалентное целое число.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
HexStringToNum_**	Группа преобразования текстовой строки с шестнадцатеричными цифрами в число	FUN	 "**" — целочисленный тип данных.	Out:=HexStringToNum_**(In); "**" — целочисленный тип данных.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Текстовая строка с шестнадцатеричными цифрами	Вход	Текстовая строка с шестнадцатеричными цифрами	Зависит от типа данных.	---	"
Out	Целое число	Выход	Целое число	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Out						OK	OK	OK	OK	OK	OK	OK	OK							

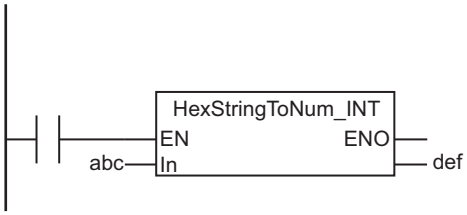
Функция

Команда HexStringToNum_** преобразует содержащуюся в *In* текстовую строку с символами шестнадцатеричных цифр в эквивалентное целое число. Любые пробелы (16#20) или «0» (16#30) в старших разрядах игнорируются. Подчеркивания (16#5F) в текстовой строке игнорируются.

Имя команды определяется типом данных *Out*. Например, если *Out* относится к типу данных INT, команда будет иметь имя HexStringToNum_INT.

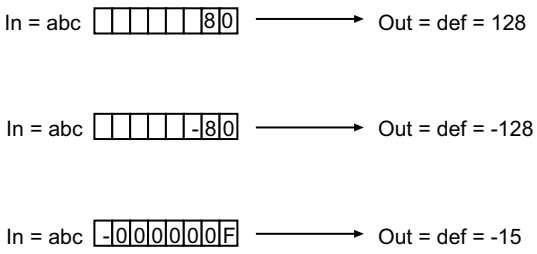
Ниже приведено несколько примеров.

LD



ST

```
def:=HexStringToNum_INT(abc);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

- Даже если результат преобразования превысит диапазон допустимых значений *Out*, ошибки не произойдет. *Out* будет содержать недопустимое значение.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Строка *In* включает символы, которые не могут быть преобразованы в числа.

FixNumToString

Команда FixNumToString преобразует число с фиксированной запятой со знаком в текстовую строку, содержащую представление этого числа в десятичном виде.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FixNumToString	Преобразование числа с фиксированной запятой в текстовую строку	FUN	(@)FixNumToString EN — ENO In Zero — Out	Out:=FixNumToString(In, Zero);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Число с фиксированной запятой	Вход	Число с фиксированной запятой со знаком	Зависит от типа данных.	---	0
Zero	Добавление нулей		Добавление нулей, если дробная часть содержит меньше 3 знаков ИСТИНА: добавлять «0» ЛОЖЬ: не добавлять «0»			ИСТИНА
Out	Текстовая строка с десятичным числом	Выход	Текстовая строка с десятичным числом	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In				OK																
Zero	OK																			
Out																				OK

Функция

Команда FixNumToString преобразует переданное в *In* число с фиксированной запятой со знаком в текстовую строку, содержащую представление этого числа в десятичном виде. Преобразование выполняется следующим образом:

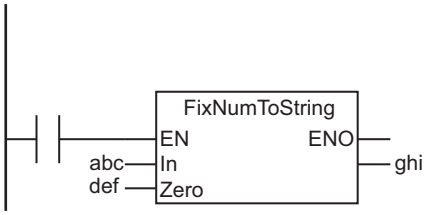
1 Содержащееся в переменной *In* шестнадцатеричное число преобразуется в десятичное число.

2 Результат делится на 1000.

Параметр *Zero* (добавление нулей) указывает, следует ли добавлять символы «0» в дробную часть числа в строке *Out*, чтобы число содержало три знака после запятой, когда число в *In* содержит два или меньше знаков после запятой. Нули добавляются, если *Zero* = ИСТИНА.
В конец строки *Out* добавляется нулевой символ (NULL).

Ниже приведено несколько примеров.

LD



ST

```
ghi:=FixNumToString(abc, def);
```

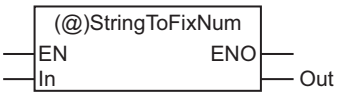
In = abc	Out = ghi	
	Zero = def = ИСТИНА	Zero = def = ЛОЖЬ
16#0001462C (10#83500)	«83.500»	«83.5»
16#00051AA4 (10#334500)	«334.500»	«334.5»
16#0003BEFC (10#245500)	«245.500»	«245.5»

Дополнительная информация

Используемый формат десятичных чисел с фиксированной запятой соответствует формату выходных значений датчиков технического зрения серии FZ компании OMRON.

StringToFixNum

Команда StringToFixNum преобразует текстовую строку с десятичным числом в число с фиксированной запятой со знаком.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
StringToFixNum	Преобразование текстовой строки в число с фиксированной запятой	FUN		Out:=StringToFixNum(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Текстовая строка с десятичным числом	Вход	Текстовая строка с десятичным числом	Зависит от типа данных.	---	''
Out	Число с фиксированной запятой	Выход	Число с фиксированной запятой	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Out				OK																

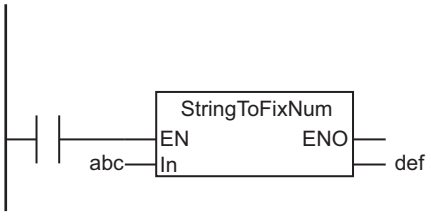
Функция

Команда StringToFixNum преобразует переданную в *In* текстовую строку, содержащую представление числа в десятичном виде, в число с фиксированной запятой со знаком. Преобразование выполняется следующим образом:

1. Число, представленное в строке *In*, умножается на 1000.
2. Дробная часть отбрасывается.
3. Полученный результат представляется в виде 32-разрядного шестнадцатеричного числа (DWORD).

Ниже приведено несколько примеров.

LD



ST

```
def:=StringToFixNum(abc);
```

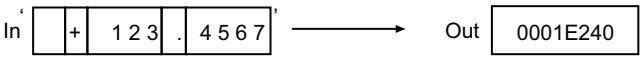
In = abc	Out = def
«83.5»	16#0001462C (10#83500)
«334.5»	16#00051AA4 (10#334500)
«245.5»	16#0003BEFC (10#245500)

На рисунке ниже показан формат текстовой строки в *In*.

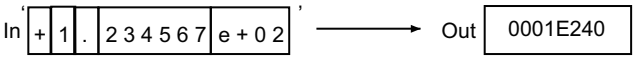
Имя	Формат
Знак	• Идущие подряд символы пробела (16#20), расположенные в начале текстовой строки, игнорируются. Если за ними следует одиночный знак плюса (+) или минуса (-), он воспринимается как знак. • Знак может быть опущен. • Идущие подряд символы пробела, расположенные после знака, игнорируются.
Целая часть	• Цифры (от «0» до «9»), располагающиеся между знаком и десятичной точкой, считаются целой частью. Знак может быть опущен. Допускается наличие пробелов между знаком и целой частью. • Если десятичная точка и дробная часть опущены, то за целую часть принимаются символы вплоть до показателя степени. • Если опущены десятичная точка, дробная часть и показатель степени, то за целую часть принимаются символы вплоть до конца текстовой строки. • Целая часть не может быть опущена. • Максимальное количество цифр в целой части определяется как максимальная длина текстовой строки (1986) минус общее количество байтов, занимаемое следующим: знак, десятичная точка, дробная часть, показатель степени и пробелы до и после знака.
Десятичная точка	• Одиночная точка («.»), следующая за целой частью, принимается за десятичную точку. • При отсутствии дробной части десятичную точку следует опустить.
Дробная часть	• Цифры (от «0» до «9»), располагающиеся между десятичной точкой и показателем степени, считаются целой частью. • Если показатель степени опущен, за дробную часть принимаются символы вплоть до конца текстовой строки. • Дробная часть может быть опущена. При отсутствии десятичной точки также отсутствует и дробная часть. • Дробная часть может состоять максимум из 15 цифр.

Имя	Формат
Показатель степени	• Показатель степени состоит из одного символа «e» или «E» после дробной части, следующего за ним одного знака «плюс» (+) или «минус» (-) и всех оставшихся цифр (от «0» до «9») до конца текстовой строки. • При отсутствии дробной части за показатель степени принимается описанная выше текстовая строка, расположенная после десятичной точки. • Если в строке нет ни дробной части, ни десятичной точки, за показатель степени принимается описанная выше текстовая строка, следующая за целой частью. • Показатель степени можно опускать. • Числовая часть показателя степени может состоять максимум из трех цифр.

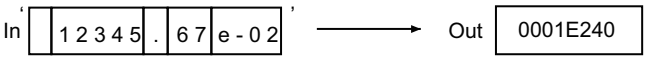
Пример 1. В следующем примере используются знак, десятичная точка и дробная часть, а показатель степени не используется.



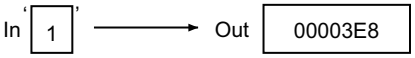
Пример 2. В следующем примере используются знак, десятичная точка, дробная часть и показатель степени.



Пример 3. В следующем примере не используется знак, но используются десятичная точка, дробная часть и показатель степени.



Пример 4. В следующем примере не используются знак, дробная часть, десятичная точка и показатель степени.



Дополнительная информация

Используемый формат десятичных чисел с фиксированной запятой соответствует формату выходных значений датчиков технического зрения серии FZ компании OMRON.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Значение в *In* усекается до трех знаков после запятой.
- Символы подчеркивания (16#5F) в текстовой строке в *In* игнорируются.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Строка *In* включает символы, которые не могут быть преобразованы в числа.
  b) В текстовой строке в *In* есть десятичная точка, но нет дробной части.

DtToString

Команда DtToString преобразует дату и время в текстовую строку.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DtToString	Преобразование даты и времени в текстовую строку	FUN	(@)DtToString EN ENO In Out	Out:=DtToString(In);

Переменные

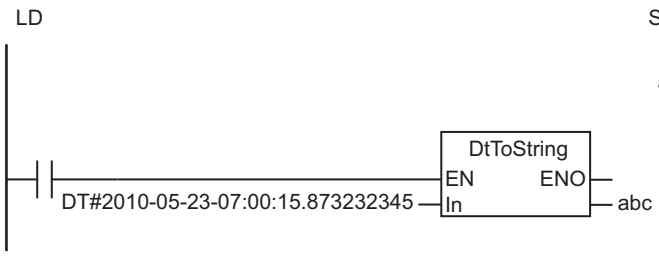
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Дата и время	Вход	Дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	DT#1970-1-1-0:0:0
Out	Текстовая строка	Выход	Текстовая строка	30 байт (29 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																			OK	
Out																				OK

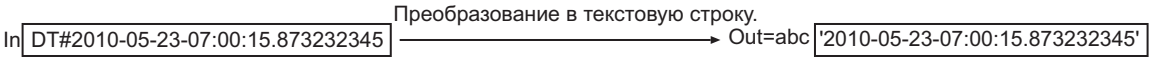
Функция

Команда DtToString преобразует дату и время в переменной *In* в текстовую строку. В конец текстовой строки *Out* добавляется нулевой символ (NULL).

В показанном ниже примере программы *In* = 2010-5-23-07:00:15.873232345 (7:00 утра + 15,873232345 сек, 23 мая 2010 г.).
Значение переменной *abc* будет равно «2010-05-23-07:00:15.873232345».



Команда DtToString преобразует дату и время ***In*** в текстовую строку.
Значение ***In*** равно 7:00 утра и 15,873232345 сек 23 мая 2010 г., поэтому значение ***abc*** будет равно «2010-05-23-07:00:15.873232345».

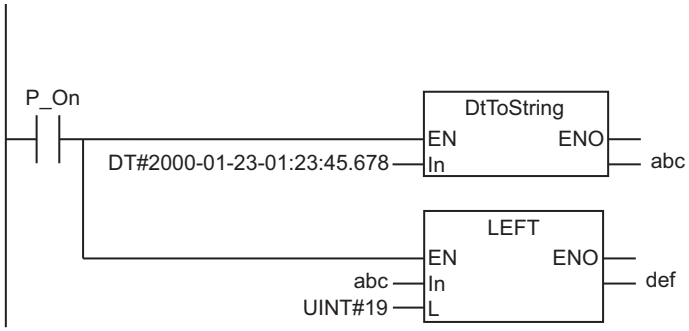


Дополнительная информация

Время в строке *Out* указывается с точностью до наносекунд. Для получения текстовой строки с указанием секунд или миллисекунд используйте эту команду совместно с командами *LEFT и RIGHT* на стр. 2-628.

Ниже приведен пример получения текстовой строки с точностью до секунд.

- LD

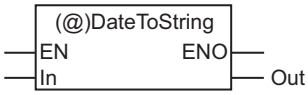


- ST

```
def:=LEFT(DtToString(DT#2000-01-23-01:23:45.678), UINT#19);
```

DateToString

Команда DateToString преобразует дату в текстовую строку.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DateToString	Преобразование даты в текстовую строку	FUN		Out:=DateToString(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Дата	Вход	Дата	Зависит от типа данных.	Год, месяц, день	DT#1970-1-1
Out	Текстовая строка	Выход	Текстовая строка	11 байт (10 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	---

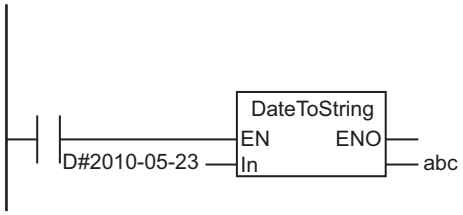
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																	OK			
Out																				OK

Функция

Команда DateToString преобразует дату в переменной *In* в текстовую строку. В конец строки *Out* добавляется нулевой символ (NULL).

В показанном ниже примере программы *In* = 2010-5-23 (23 мая 2010 г.).
Значение переменной *abc* будет равно «2010-05-23».

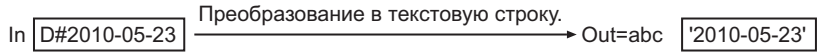
LD



ST

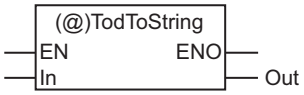
```
abc:=DateToString(D#2010-05-23);
```

Команда DateToString преобразует дату ***In*** в текстовую строку.
Значение ***In*** равно 23 мая 2010 г., поэтому значение ***abc*** будет равно «2010-05-23».



TodToString

Команда TodToString преобразует время суток в текстовую строку.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TodToString	Преобразование времени суток в текстовую строку	FUN		Out:=TodToString(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Время суток	Вход	Время суток	Зависит от типа данных.	Часы, минуты, секунды	TOD#0:0:0
Out	Текстовая строка	Выход	Текстовая строка	19 байт (18 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	---

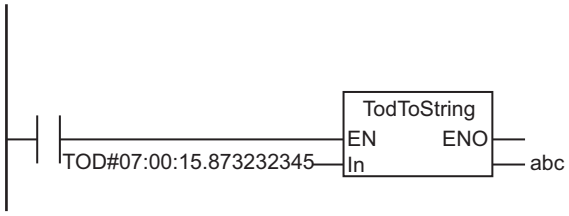
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																		OK		
Out																				OK

Функция

Команда TodToString преобразует время суток в переменной *In* в текстовую строку. В конец строки *Out* добавляется нулевой символ (NULL).

В показанном ниже примере программы *In* = 07:00:15.873232345 (7:00 утра + 15,873232345 сек). Значение переменной *abc* будет равно «07:00:15.873232345».

LD



ST

```
abc:=TodToString(TOD#07:00:15.873232345);
```

Команда TodToString преобразует время суток ***In*** в текстовую строку.
Значение ***In*** равно 7:00 утра и 15,873232345 сек, поэтому значение ***abc*** будет равно «07:00:15.873232345».

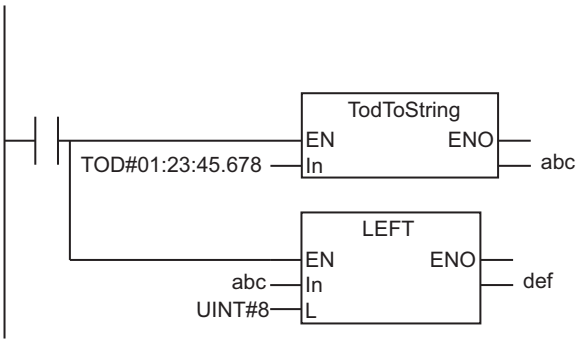
In TOD#07:00:15.873232345 → Преобразование в текстовую строку. → Out=abc '07:00:15.873232345'

Дополнительная информация

Время в строке *Out* указывается с точностью до наносекунд.
Для получения текстовой строки с указанием секунд или миллисекунд используйте эту команду совместно с командами *LEFT и RIGHT* на стр. 2-628.

Ниже приведен пример получения текстовой строки с точностью до секунд.

• LD

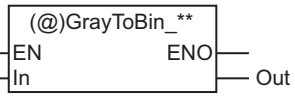
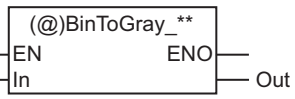


• ST

```
def:=LEFT(TodToString(TOD#01:23:45.678), UINT#8);
```

GrayToBin_** и BinToGray_**

GrayToBin_** : Преобразует код Грея в битовую строку.
BinToGray_** : Преобразует битовую строку в код Грея.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GrayToBin_**	Группа преобразования кода Грея в двоичный код	FUN	 "**" — тип данных, относящийся к битовым строкам.	Out:=GrayToBin_**(In); "**" — тип данных, относящийся к битовым строкам.
BinToGray_**	Преобразование двоичного кода в код Грея	FUN	 "**" — тип данных, относящийся к битовым строкам.	Out:=BinToGray_**(In); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	0
Out	Результат преобразования	Выход	Результат преобразования	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		ОК	ОК	ОК	ОК															
Out	Тип данных должен быть таким же, как у *In*.																			

Функция

Имена команд определяются типами данных *In* и *Out*. Например, если *In* и *Out* относятся к типу данных WORD, команды будут иметь имена GrayToBin_WORD и BinToGray_WORD.

GrayToBin_**

Команды GrayToBin_** преобразуют код Грея в *In* (данные для преобразования) в битовую строку.

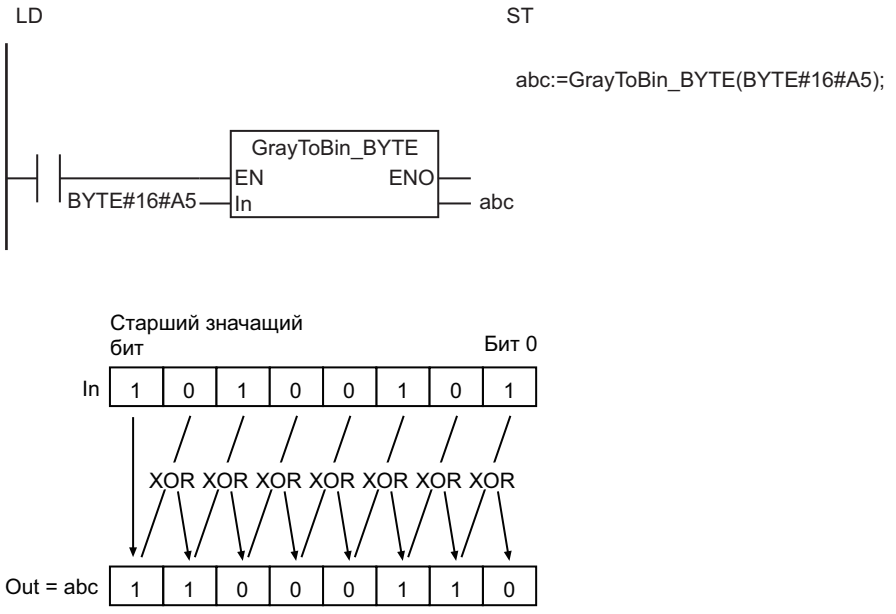
Если *In* и *Out* имеют тип данных BYTE, преобразование выполняется следующим образом:

1 Значение самого старшего бита (бита 7) *In* присваивается самому старшему биту (биту 7) *Out*.

2 Результат операции исключающего логического ИЛИ над битом 6 *In* и битом 7 *Out* присваивается биту 6 *Out*.

3 Этот процесс повторяется до достижения самого младшего бита (бита 0) *Out*.

Ниже показан пример для команды GrayToBin_BYTE, в котором *In* = BYTE#16#A5.



BinToGray_**

Команда BinToGray_** преобразует битовую строку в *In* (данные для преобразования) в код Грея.

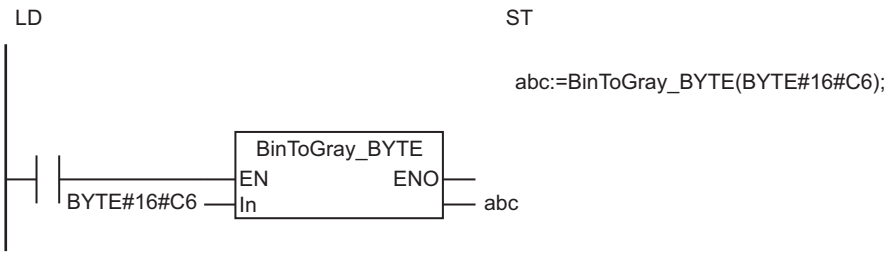
Если *In* и *Out* имеют тип данных BYTE, преобразование выполняется следующим образом:

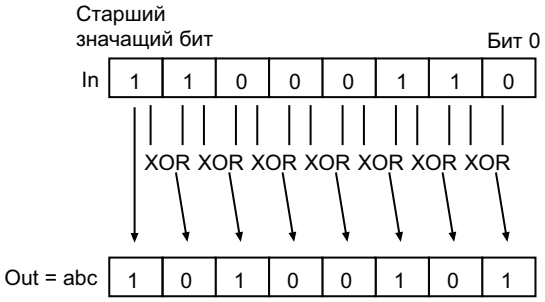
1 Значение самого старшего бита (бита 7) *In* присваивается самому старшему биту (биту 7) *Out*.

2 Результат операции исключающего логического ИЛИ над битом 7 *In* и битом 6 *In* присваивается биту 6 *Out*.

3 Этот процесс повторяется до достижения самого младшего бита (бита 0) *Out*.

Ниже показан пример для команды BinToGray_BYTE, в котором *In* = BYTE#16#C6.



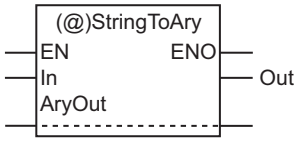


Меры предосторожности для обеспечения надлежащей эксплуатации

Типы данных *In* и *Out* должны быть одинаковыми.

StringToAry

Команда StringToAry преобразует текстовую строку в массив байтов (BYTE).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
StringToAry	Преобразование текстовой строки в массив	FUN		Out:=StringToAry(In, AryOut);

Переменные

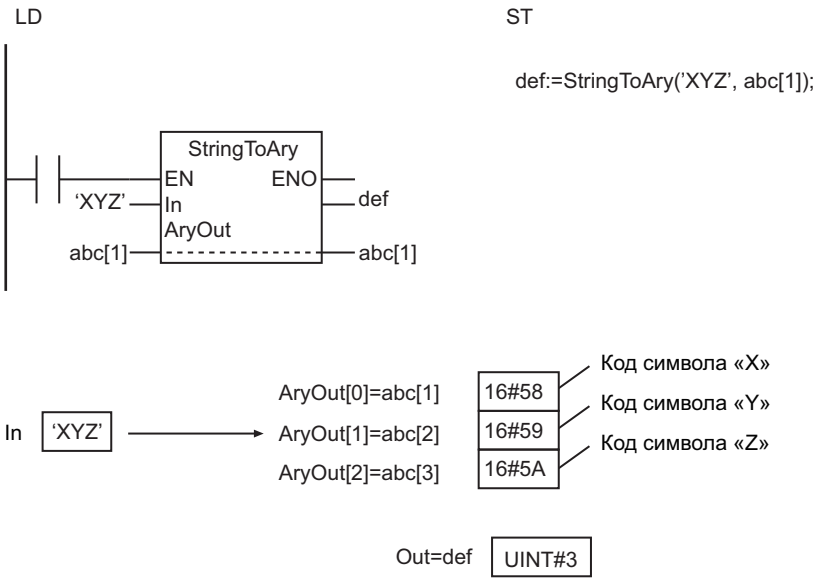
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Текстовая строка	Вход	Текстовая строка	Зависит от типа данных.	---	''
AryOut[] (массив)	Массив типа BYTE	Вход-выход	Массив типа BYTE	Зависит от типа данных.	---	---
Out	Количество байтов для преобразования	Выход	Количество байтов для преобразования	0...1985	Байты	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
AryOut[] (массив)		OK																		
Out							OK													

Функция

Команда StringToAry обрабатывает коды символов в текстовой строке *In* как числовые значения и присваивает каждое числовое значение соответствующему элементу массива AryOut [] типа BYTE.
Количество преобразованных байтов сохраняется в *Out*.

В показанном ниже примере программы *In* = «XYZ».

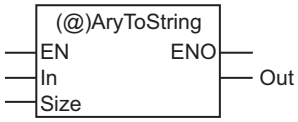


Меры предосторожности для обеспечения надлежащей эксплуатации

- Нулевой символ (NULL) в конце строки *In* в массив AryOut[] не сохраняется.
- Если текстовая строка *In* содержит только нулевые символы (NULL), то значение *Out* будет равно 0, а содержимое AryOut[] не изменится.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значение *Out* и содержимое AryOut[] не изменятся.
  a) Количество байтов в *In* больше, чем количество элементов в массиве AryOut[].

AryToString

Команда AryToString преобразует массив байтов (BYTE) в текстовую строку.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryToString	Преобразование массива в текстовую строку	FUN		Out:=AryToString(In, Size);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив типа BYTE	Вход	Массив типа BYTE Максимальное количество элементов: 1985	Зависит от типа данных.	---	*1
Size	Количество элементов для преобразования		Количество элементов In[] для преобразования	0...1985		1
Out	Текстовая строка	Выход	Текстовая строка	Зависит от типа данных.	---	---

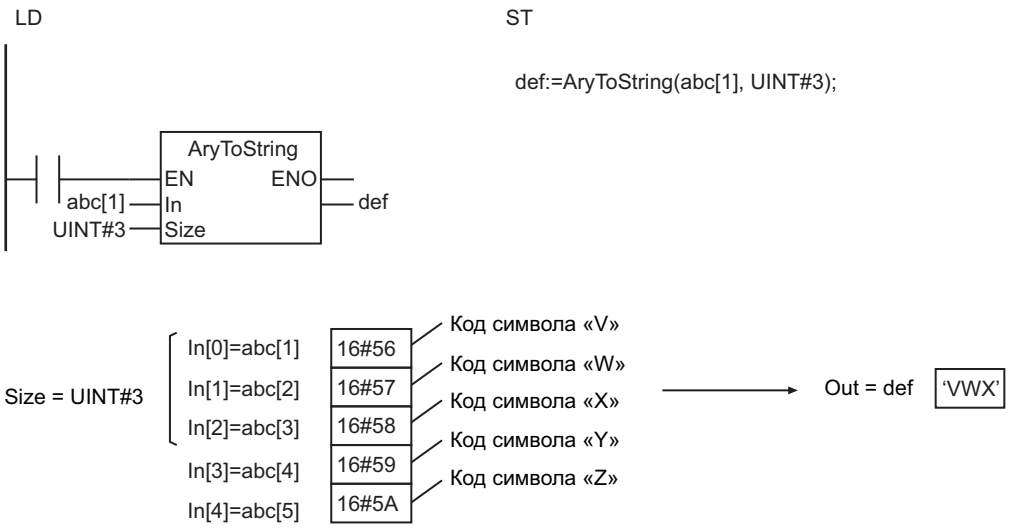
*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)		OK																		
Size							OK													
Out																				OK

Функция

Команда AryToString обрабатывает элементы массива In[] типа BYTE (начиная с элемента In[0]) как коды символов и преобразует их в текстовую строку, которая сохраняется в *Out*.
В конец строки *Out* добавляется нулевой символ (NULL).
Количество преобразуемых элементов в массиве In[] указывается параметром *Size*. Если какой-либо из элементов между In[0] и In[*Size*-1] содержит нулевой символ (NULL), в переменную *Out* сохраняются только коды символов, расположенных до этого нулевого символа.

В показанном ниже примере программы *Size* = UINT#3.

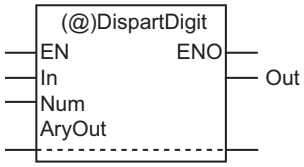


Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *Size* = 0, то *Out* представляет собой текстовую строку, содержащую только нулевые символы (NULL).
- В указанном ниже случае происходит ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Size* приводит к выходу за область массива In[].

DispartDigit

Команда DispartDigit разделяет битовую строку на 4-битные блоки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DispartDigit	Разделение на 4-битные блоки	FUN		DispartDigit(In, Num, AryOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для разделения	Вход	Битовая строка для разделения	Зависит от типа данных.	---	*1
Num	Количество разрядов для разделения		Количество разрядов для разделения	От 0 до количества битов в *In*		1
AryOut[] (массив)	Массив с результатами разделения	Вход-выход	Массив с результатами разделения	16#00...16#0F	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Num						OK														
AryOut[] (массив)		OK																		
Out	OK																			

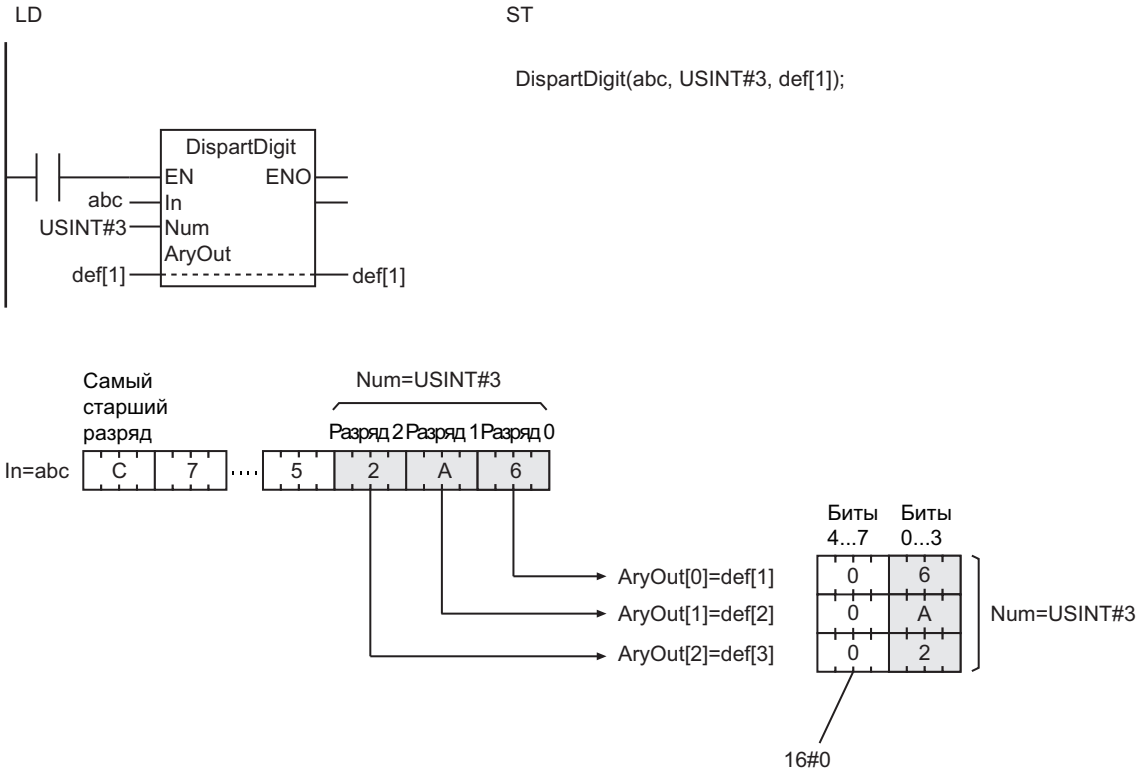
Функция

Команда DispartDigit разделяет значение в *In* (данные для разделения) на блоки длиной 4 бита (шестнадцатеричные разряды) и сохраняет их в массив AryOut[] (результаты разделения).

Сначала переданное в *In* значение разделяется на тетрады (блоки по 4 бита). Затем первая тетрада сохраняется в четыре младших бита элемента AryOut[0]. AryOut[0] имеет тип данных BYTE, и в его биты 4...7 записывается значение 16#0.

Этот процесс повторяется *Num* (количество разрядов для разделения) раз.

Ниже показан пример программы, в котором *Num* = USINT#3.



Дополнительная информация

Для объединения отдельных 4-битных блоков в единую битовую строку используйте команду *UniteDigit_*** на стр. 2-535.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *Num* = 0, значения в массиве AryOut[] не изменяются.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение *Num* находится за пределами допустимого диапазона.
  b) Значение *Num* приводит к выходу за область массива AryOut[].

UniteDigit_**

Команды UniteDigit_** объединяют 4-битные блоки данных в битовую строку.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
UniteDigit_**	Группа объединения 4-битных блоков	FUN	(@)UniteDigit_** EN ENO In Out Num "**" — тип данных, относящийся к битовым строкам.	Out:=UniteDigit_**(In, Num); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для объединения	Вход	Массив для объединения	Зависит от типа данных.	---	*1
Num	Количество разрядов для объединения		Количество разрядов для объединения	От 0 до количества битов в *Out*		1
Out	Результат объединения	Выход	Битовая строка с результатом объединения	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)		ОК																		
Num						ОК														
Out		ОК	ОК	ОК	ОК															

Функция

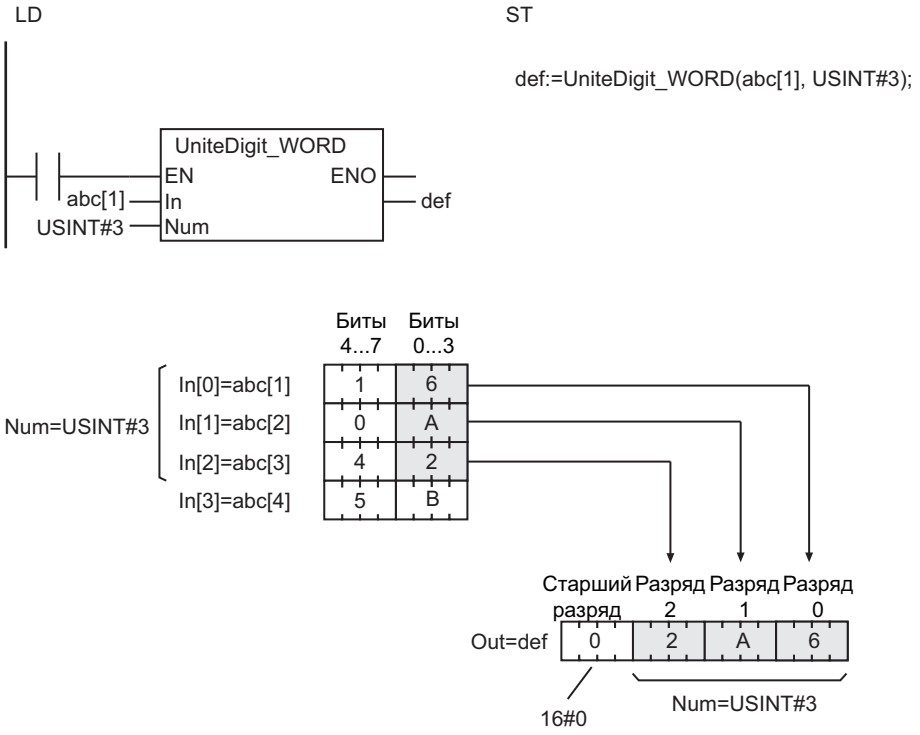
Команды UniteDigit_** объединяют блоки из четырех младших битов (тетрады, т. е. шестнадцатеричные разряды) каждого элемента массива In[] (массив для объединения) в единую битовую строку, которая сохраняется в *Out* (результат объединения).

Количество элементов массива, тетрады которых должны быть объединены, задается параметром *Num* (количество разрядов для объединения).

Сначала младшие тетрады (блоки из четырех младших битов) каждого элемента массива In[] (от In[0] до In[*Num*-1]) объединяются для создания битовой строки, содержащей *Num* шестнадцатеричных разрядов (цифр).
Затем к этой битовой строке в качестве старшего разряда присоединяется число 16#0 (результат вычитания *Num* из количества разрядов в *Out*) и полученная объединенная строка присваивается переменной *Out*.

Имя команды определяется типом данных *Out*. Например, если *Out* относится к типу данных WORD, команда будет иметь имя UniteDigit_WORD.

Ниже показан пример для команды UniteDigit_WORD, в котором *Num* = USINT#3.



Дополнительная информация

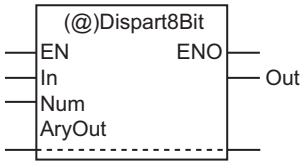
Для разделения битовой строки на 4-битные блоки используйте команду *DispartDigit* на стр. 2-533.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если значение *Num* равно 0, то значение *Out* также равно 0.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Num* находится за пределами допустимого диапазона.
  b) Значение *Num* приводит к выходу за область массива In[].

Dispart8Bit

Команда Dispart8Bit разделяет битовую строку на отдельные байты.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Dispart8Bit	Разделение данных на байты	FUN		Dispart8Bit(In, Num, AryOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для разделения	Вход	Битовая строка для разделения	Зависит от типа данных.	---	*1
Num	Количество байтов для разделения		Количество байтов для разделения	От 0 до количества байтов в *In*		1
AryOut[] (массив)	Массив с результатами разделения	Вход-выход	Массив с результатами разделения	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		OK	OK	OK	OK															
Num						OK														
AryOut[] (массив)		OK																		
Out	OK																			

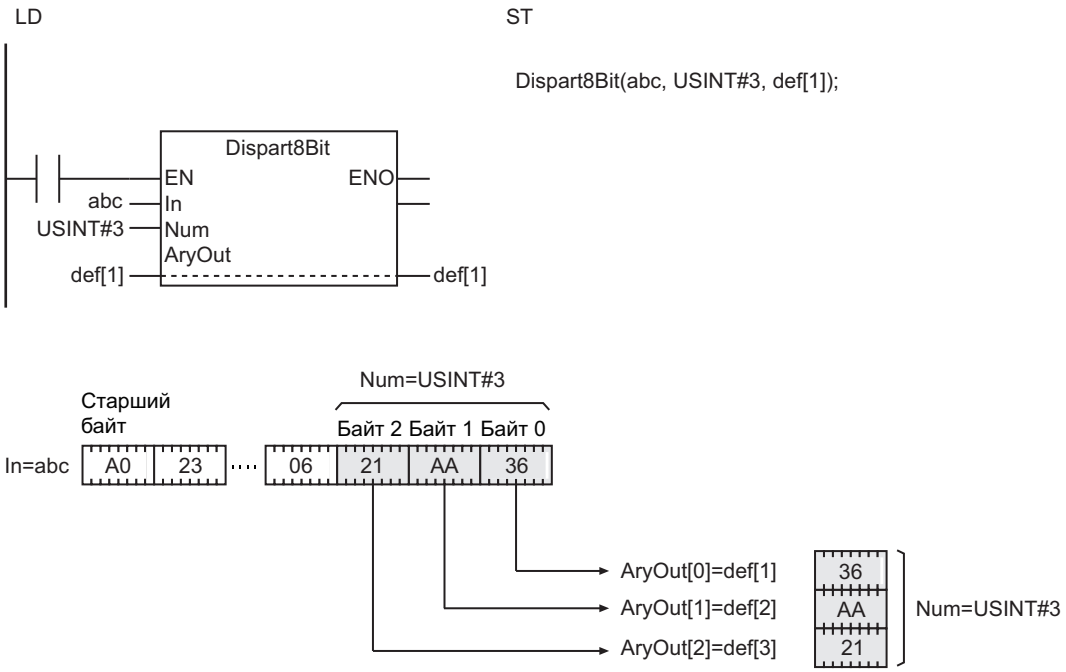
Функция

Команда Dispart8Bit разделяет значение в *In* (данные для разделения) на отдельные байты и сохраняет их в массив AryOut[] (массив с результатами разделения).

Сначала переданное в *In* значение разделяется на байты. Самый младший байт сохраняется в элемент AryOut[0].

Следующий по порядку младший байт сохраняется в элемент AryOut[1]. Этот процесс повторяется *Num* (количество байтов для разделения) раз.

Ниже показан пример программы, в котором *Num* = USINT#3.



Дополнительная информация

Для объединения отдельных байтов (однобайтных элементов массива) в единую битовую строку используйте команду *Unite8Bit_*** на стр. 2-539.

Меры предосторожности для обеспечения надлежащей эксплуатации

- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое AryOut[] не изменится.
  a) Значение *Num* находится за пределами допустимого диапазона.
  b) Значение *Num* приводит к выходу за область массива AryOut[].

Unite8Bit_**

Команды Unite8Bit_** объединяют байты данных в битовую строку.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Unite8Bit_**	Группа объединения байтов данных	FUN	 "**" — тип данных, относящийся к битовым строкам.	Out:=Unite8Bit_**(In, Num); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для объединения	Вход	Массив для объединения	Зависит от типа данных.	---	*1
Num	Количество байтов для объединения		Количество байтов для объединения	От 0 до количества байтов в *Out*		1
Out	Результат объединения	Выход	Битовая строка с результатом объединения	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)		ОК																		
Num						ОК														
Out		ОК	ОК	ОК	ОК															

Функция

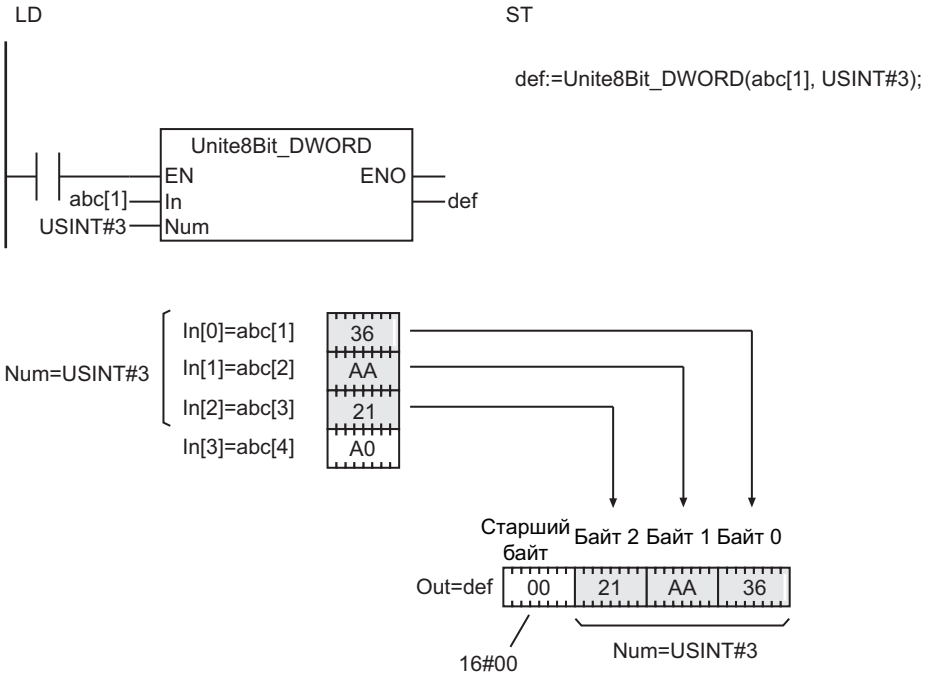
Команды Unite8Bit_** объединяют элементы массива In[] (массив для объединения) в единую битовую строку, которая сохраняется в *Out* (результат объединения).

Количество элементов массива, которые должны быть объединены, задается параметром *Num* (количество байтов для объединения). Сначала элементы массива с In[0] по In[*Num*-1] объединяются для создания битовой строки, содержащей *Num* байтов.

Затем к этой битовой строке в качестве старшего байта присоединяется число 16#0 (результат вычитания *Num* из количества байтов в *Out*) и полученная объединенная строка присваивается переменной *Out*.

Имя команды определяется типом данных *Out*. Например, если *Out* относится к типу данных DWORD, команда будет иметь имя Unite8Bit_DWORD.

Ниже показан пример работы команды Unite8Bit_DWORD для случая, когда *Num* = USINT#3.



Дополнительная информация

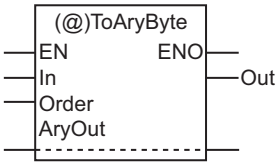
Для разделения битовой строки на однобайтные блоки используйте команду *Dispart8Bit* на стр. 2-537.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если значение *Num* равно 0, то значение *Out* также равно 0.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Num* находится за пределами допустимого диапазона.
  b) Значение *Num* приводит к выходу за область массива In[].

ToAryByte

Команда ToAryByte разделяет переменную на байты и сохраняет их в массив типа BYTE.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ToAryByte	Преобразование в байтовый массив	FUN		Out:=ToAryByte(In, Order, AryOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Данные для преобразования	Зависит от типа данных.	---	*1
Order	Порядок преобразования		Порядок преобразования	_LOW_HIGH или _HIGH_LOW		_LOW_HIGH
AryOut[] (массив)	Массив результатов преобразования	Вход-выход	Массив результатов преобразования	Зависит от типа данных.	---	---
Out	Количество элементов в результате	Выход	Количество элементов в результате	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать перечисление, массив, элемент массива, структуру или член структуры.																			
Order	Сведения о перечислителях перечислимого типа _eBYTE_ORDER см. в разделе *Функция* на стр. 2-541.																			
AryOut[] (массив)		OK																		
Out							OK													

Функция

Команда ToAryByte разделяет значение *In* на отдельные байты и сохраняет их по порядку в массив AryOut[] (массив результатов преобразования), начиная с элемента AryOut[0].

Переменная *Out* (количество элементов в результате) содержит количество элементов, хранящихся в AryOut[].

Порядок, в котором значение *In* преобразуется в байты, задается параметром *Order*. Для параметра *Order* используется перечислимый тип данных _eBYTE_ORDER. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_LOW_HIGH	Младший байт первым, старший байт последним
_HIGH_LOW	Старший байт первым, младший байт последним

Если длина типа данных *In* два байта или больше

Если длина типа данных переменной *In* составляет два байта или больше, *In* разделяется на байты, которые сохраняются в AryOut[].
Ниже перечислены типы данных, длина которых составляет 2 байта или больше.

Классификация	Тип данных
Битовые строки	WORD, DWORD и LWORD
Целочисленные типы	UINT, UDINT, ULINT, INT, DINT и LINT
Вещественные типы	REAL и LREAL
Значения времени и продолжительности, даты и текстовые строки	Типы данных TIME, DATE, TOD, DT и STRING длиной 2 байта или больше
Прочее	• Перечисление • Массив, общая длина всех элементов которого составляет 2 байта или больше • Элемент массива размером 2 байта или более • Структура, общая длина всех членов которой составляет 2 байта или больше • Член структуры размером 2 байта или более

Обработка выполняется в следующем порядке:

1 Сначала значение в *In* разделяется на слова (1 слово = 2 байта).

2 Самое младшее слово разделяется на байты.

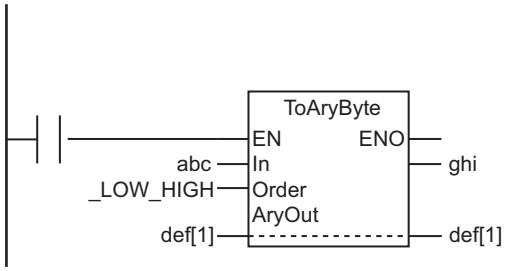
3 Если *Order* = _LOW_HIGH, то младший байт сохраняется в AryOut[0], а старший байт — в AryOut[1]. Если *Order* = _HIGH_LOW, то старший байт сохраняется в AryOut[0], а младший байт — в AryOut [1].

4 Затем следующее слово разделяется на байты, которые аналогичным образом сохраняются в элементы AryOut[2] и AryOut[3].

5 Этот процесс повторяется до достижения последнего байта в *In*. Если *In* является массивом, то этот же процесс повторяется до достижения последнего элемента *In*.

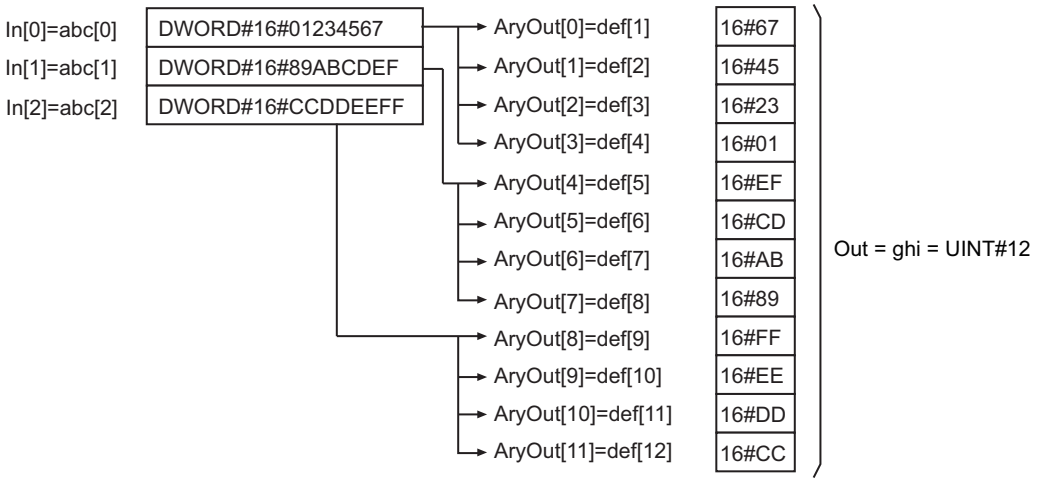
Ниже представлен пример, в котором *In* является массивом типа DWORD с тремя элементами, а *Order* = _LOW_HIGH.

LD



ST

```
ghi:=ToAryByte(abc, _LOW_HIGH, def[1]);
```



Ниже показан пример с таким же значением *In*, как и в примере выше, и *Order* = _HIGH_LOW.

Если длина типа данных *In* один байт

Если длина типа данных переменной *In* составляет один байт, *In* сохраняется в массив AryOut[] как один байт.

Ниже перечислены типы данных, длина которых составляет 1 байт.

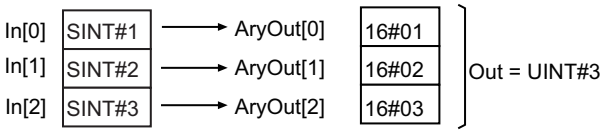
Классификация	Тип данных
Битовые строки	BYTE
Целочисленные типы	USINT и SINT
Вещественные типы	Нет

Классификация	Тип данных
Значения времени и продолжительности, даты и текстовые строки	Строковые типы (STRING) длиной в один байт
Прочее	• Массив, общая длина всех элементов которого составляет 1 байт • Элемент массива размером 1 байт • Структура, общая длина всех членов которой составляет 1 байт • Член структуры размером 1 байт

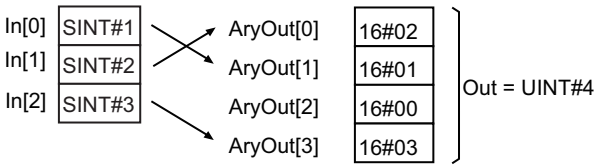
Используется один из указанных ниже способов хранения.

Значение *Order*	*In* (массив или нет)	Способ хранения в AryOut[]
_LOW_HIGH	Не массив	Значение *In* сохраняется в AryOut[0].
	Массив	Значение In[i] сохраняется в AryOut[i].
_HIGH_LOW	Не массив	Значение *In* сохраняется в AryOut[1]. 16#00 сохраняется в AryOut[0].
	Массив	In[i] (где *i* — четное) сохраняется в AryOut[i+1]. In[i] (где *i* — нечетное) сохраняется в AryOut[i-1]. Если количество элементов в In[] нечетное, то в конце процедуры в AryOut[n-1] сохраняется значение 16#00.

Ниже представлен пример, в котором *In* является массивом типа SINT с тремя элементами, а *Order* = _LOW_HIGH.



Ниже показан пример с таким же значением *In*, как и в примере выше, и *Order* = _HIGH_LOW.



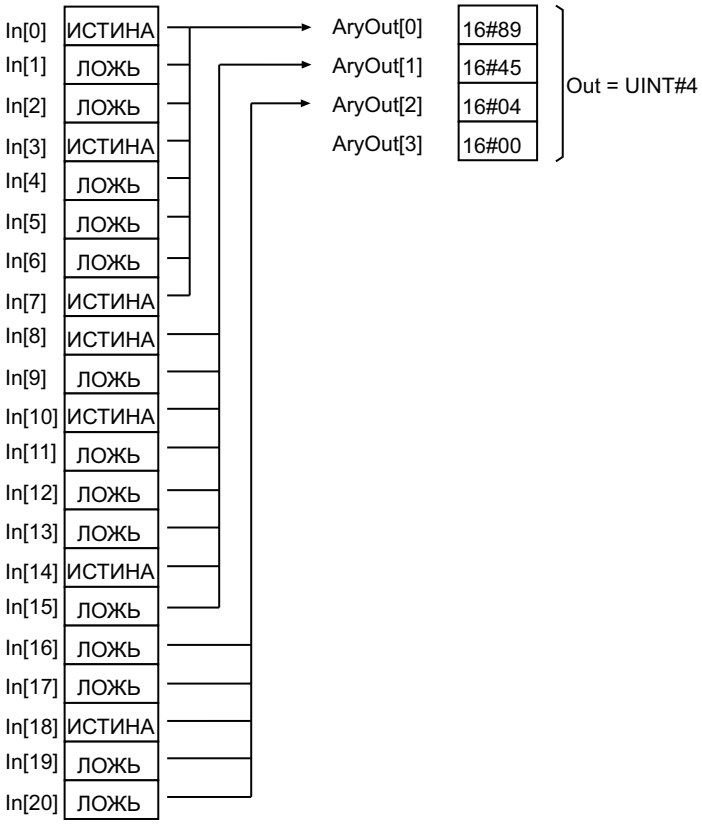
Если *In* имеет тип данных BOOL

Если переменная *In* имеет тип данных BOOL (один бит), данные сохраняются в массив AryOut[] указанным ниже образом.

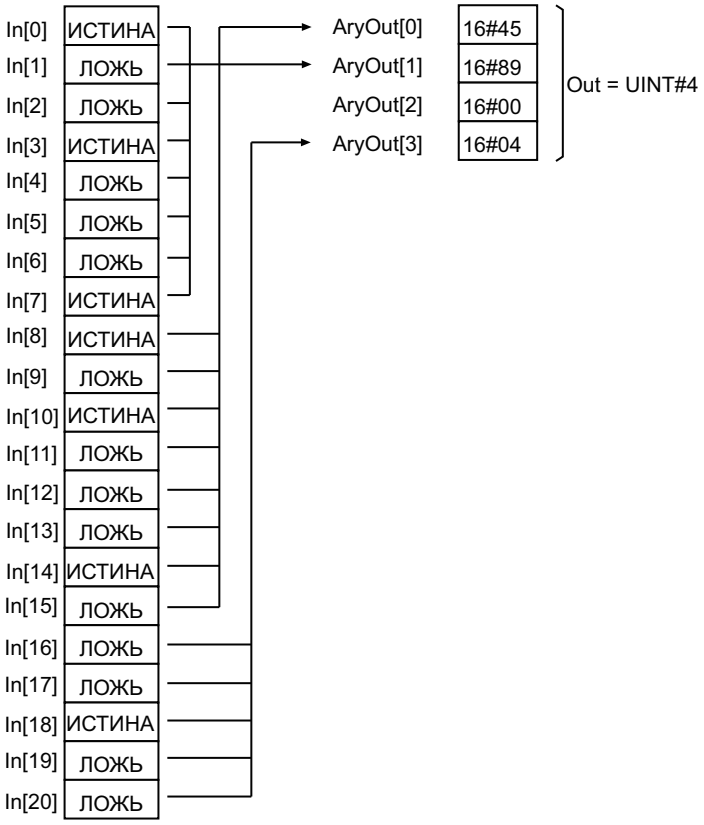
Значение *Order*	*In* (массив или нет)	Способ хранения в AryOut[]
_LOW_HIGH	Не массив	Результат логического ИЛИ значения *In* и 16#00 сохраняется в AryOut[0].
	Массив	Значения In[0]...In[7] объединяются и сохраняются в AryOut[0]. Значения In[8]...In[15] объединяются и сохраняются в AryOut[1]. Аналогичным образом сохраняются все остальные данные. Если массив In[] содержит меньше 8 значений, то в старший значащий бит добавляется значение ЛОЖЬ. Таким образом, значение *Out* всегда четное. Если количества битов недостаточно, то все остальные биты результата будут содержать ЛОЖЬ.

Значение *Order*	*In* (массив или нет)	Способ хранения в AryOut[]
_HIGH_LOW	Не массив	Результат логического ИЛИ значения *In* и 16#00 сохраняется в AryOut[1]. 16#00 сохраняется в AryOut[0].
	Массив	Значения In[0]...In[7] объединяются и сохраняются в AryOut[1]. Значения In[8]...In[15] объединяются и сохраняются в AryOut[0]. Аналогичным образом сохраняются все остальные данные. Таким образом, значение *Out* всегда четное. Если количества битов недостаточно, то все остальные биты результата будут содержать ЛОЖЬ.

В представленном ниже примере *In* является массивом типа BOOL с 21 элементом, а *Order* = _LOW_HIGH.



В приведенном ниже примере *In* содержит то же значение, что и в примере выше, а *Order* = _HIGH_LOW.

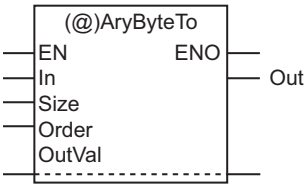


Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте переменную в качестве входного параметра, передаваемого в *In*. При передаче константы произойдет ошибка сборки.
- Если параметр *In* является перечислением, передать непосредственно перечислитель невозможно. В случае непосредственной передачи перечислителя произойдет ошибка сборки.
- Если *In* является строковым типом (STRING), текстовая строка в числа не преобразуется. Содержимое переменной воспринимается как битовая строка и преобразуется в байтовый массив.
- Если *In* является структурой, в массиве AryOut[] между членами структуры могут быть вставлены дополнительные согласующие области.
- Если *Order* = _HIGH_LOW и общее количество байтов в *In* является нечетным числом, перед преобразованием в конец *In* добавляется 16#00, чтобы число байтов стало четным.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значение *Out* и содержимое AryOut[] не изменятся.
  a) Значение *Order* находится за пределами допустимого диапазона.
  b) Результат преобразования выходит за область массива AryOut[].

AryByteTo

Команда AryByteTo объединяет элементы массива типа BYTE и сохраняет результат в переменной.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryByteTo	Преобразование из байтового массива	FUN		AryByteTo(In, Size, Order, OutVal);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для преобразования	Вход	Массив для преобразования	Зависит от типа данных.	---	*1
Size	Количество элементов для преобразования		Количество элементов в In[] для преобразования			1
Order	Порядок преобразования		Порядок преобразования	_LOW_HIGH или _HIGH_LOW		_LOW_ HIGH
OutVal	Результат преобразования	Вход-выход	Результат преобразования	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)		ОК																		
Size							ОК													
Order	Сведения о перечислителях перечислимого типа _eBYTE_ORDER см. в разделе *Функция* на стр. 2-548.																			
OutVal	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также можно указать перечисление, массив, элемент массива, структуру или член структуры.																			
Out	ОК																			

Функция

Команда AryByteTo берет первые *Size* элементов массива In[] (массив для преобразования) и объединяет их, обеспечивая при этом соответствие размеру типа данных переменной *OutVal* (результат преобразования). Полученный результат сохраняется в *OutVal*.

Порядок, в котором объединяются элементы In[], задается параметром *Order*. Для параметра *Order* используется перечислимый тип данных _eBYTE_ORDER. Значения перечислителей приведены в таблице ниже.

Перечислители	Значение
_LOW_HIGH	Младший байт первым, старший байт последним
_HIGH_LOW	Старший байт первым, младший байт последним

Если длина типа данных *OutVal* два байта или больше

Если длина типа данных переменной *OutVal* составляет два байта или больше, элементы массива In[] объединяются в блок, длина которого равна длине типа данных *OutVal*, и этот блок объединенных данных сохраняется в *OutVal*.
Ниже перечислены типы данных, длина которых составляет 2 байта или больше.

Классификация	Тип данных
Битовые строки	WORD, DWORD и LWORD
Целочисленные типы	UINT, UDINT, ULINT, INT, DINT и LINT
Вещественные типы	REAL и LREAL
Значения времени и продолжительности, даты и текстовые строки	Типы данных TIME, DATE, TOD, DT и STRING длиной 2 байта или больше
Прочее	• Перечисление • Массив, общая длина всех элементов которого составляет 2 байта или больше • Элемент массива размером 2 байта или более • Структура, общая длина всех членов которой составляет 2 байта или больше • Член структуры размером 2 байта или более

Обработка выполняется в следующем порядке:

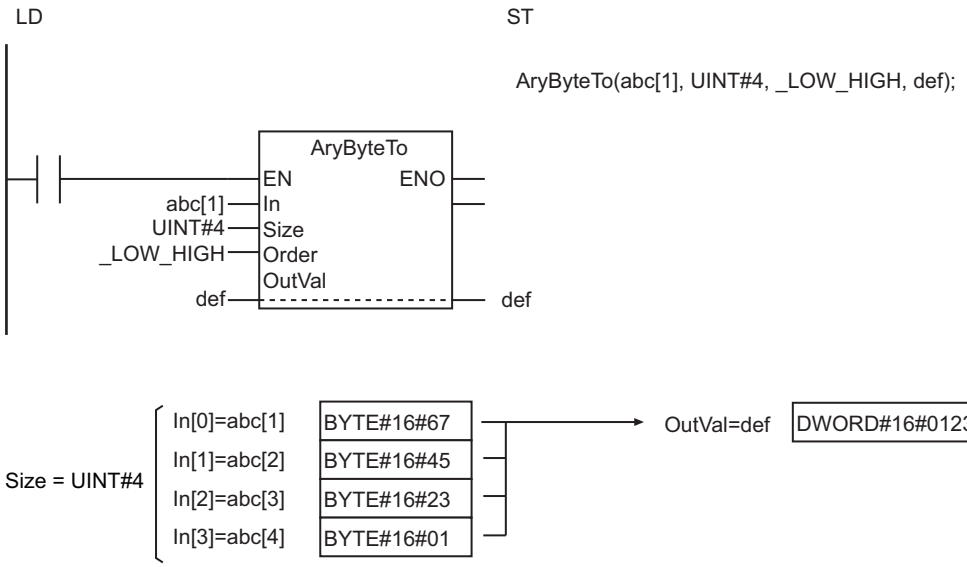
1 In[0] и In[1] объединяются в соответствии со значением *Order* для создания одного слова (двух байтов) данных.
Если *Order* = _LOW_HIGH, то старший байт сохраняется в In[1], а младший байт — в In[0].
Если *Order* = _HIGH_LOW, то старший байт сохраняется в In[0], а младший байт — в In[1].

2 Аналогичным образом объединяются остальные элементы, начиная с In[2] и In[3], и создаются другие слова данных.

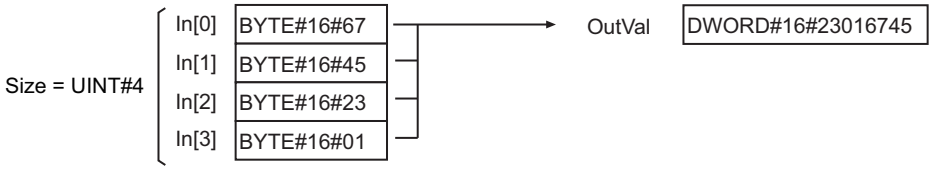
3 Слова данных объединяются с учетом размера переменной *OutVal*. Например, если *OutVal* — значение типа DWORD, то объединяются четыре слова данных.

4 Объединенные данные сохраняются в *OutVal*.

Ниже представлен пример, в котором *OutVal* является значением типа DWORD, *Size* = UINT#4, а *Order* = _LOW_HIGH.



В приведенном ниже примере *OutVal* имеет то же значение, что и в примере выше, *Size* = UINT#4, а *Order* = _HIGH_LOW.



Если длина типа данных *OutVal* один байт

Если длина типа данных переменной *OutVal* составляет один байт, то в *OutVal* сохраняется непосредственно один байт переменной In[].
Ниже перечислены типы данных, длина которых составляет 1 байт.

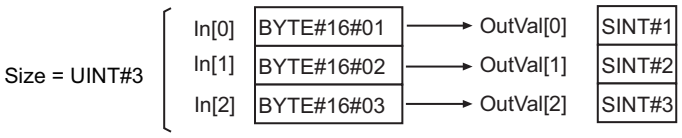
Классификация	Тип данных
Битовые строки	BYTE
Целочисленные типы	USINT и SINT
Вещественные типы	Нет
Значения времени и продолжительности, даты и текстовые строки	Строковые типы (STRING) длиной в один байт
Прочее	• Массив, общая длина всех элементов которого составляет 1 байт • Элемент массива размером 1 байт • Структура, общая длина всех членов которой составляет 1 байт • Член структуры размером 1 байт

Используется один из указанных ниже способов хранения.

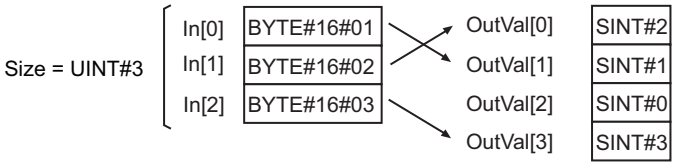
Значение *Order*	*OutVal* (массив или нет)	Способ хранения в *OutVal*
_LOW_HIGH	Не массив	Значение элемента In[0] сохраняется в *OutVal*
	Массив	Значение In[i] сохраняется в OutVal[i].

Значение *Order*	*OutVal* (массив или нет)	Способ хранения в *OutVal*
_HIGH_LOW	Не массив	Значение элемента In[1] сохраняется в *OutVal*
	Массив	In[i] (где *i* — четное) сохраняется в OutVal[i+1]. In[i] (где *i* — нечетное) сохраняется в OutVal[i-1]. Если значение *Size* нечетное, данные сохраняются в элементы вплоть до OutVal[*Size*], а в элемент OutVal[*Size*-1] записывается 16#00.

Ниже представлен пример, в котором *OutVal* является массивом типа SINT с тремя элементами, *Size* = UINT#3, а *Order* = _LOW_HIGH.



В показанном ниже примере *OutVal* и *Size* имеют те же значения, что и в примере выше, а *Order* = _HIGH_LOW.

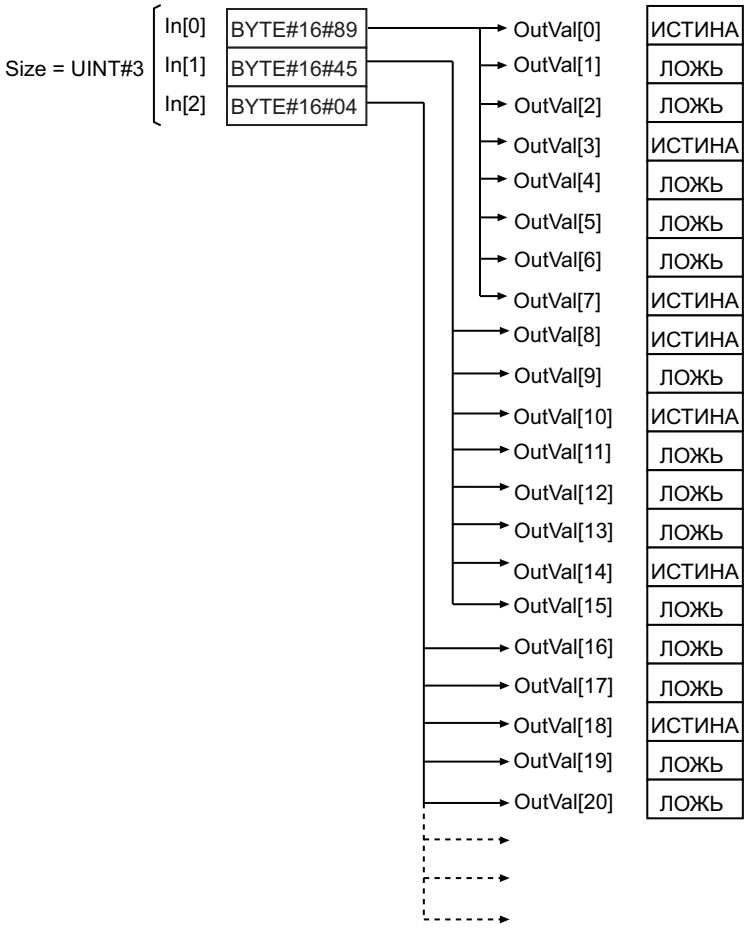


Если *OutVal* имеет тип данных BOOL

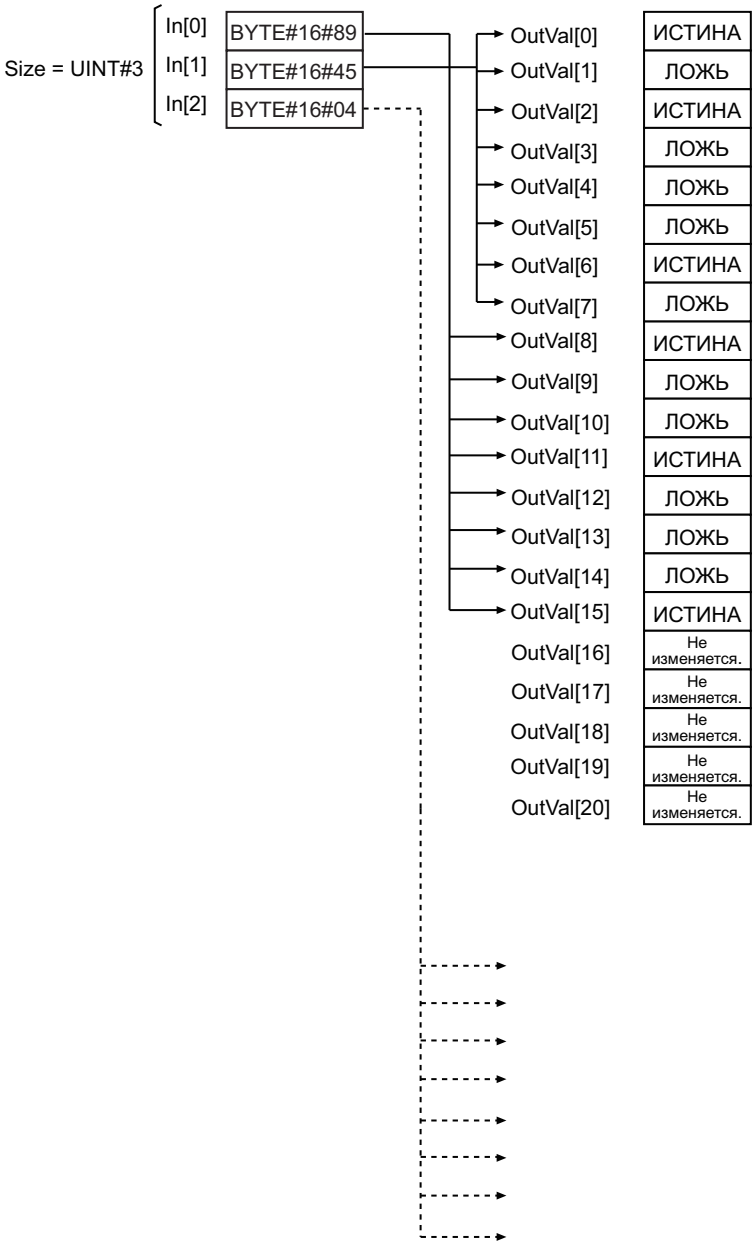
Если переменная *OutVal* имеет тип данных BOOL (один бит), данные сохраняются в *OutVal* указанным ниже образом.

Значение *Order*	*OutVal* (массив или нет)	Способ хранения в *OutVal*
_LOW_HIGH	Не массив	Значение бита 0 элемента In[0] сохраняется в *OutVal*.
	Массив	Значение элемента In[0] разделяется на биты, которые сохраняются в элементы OutVal[0]...OutVal[7]. Значение элемента In[1] разделяется на биты, которые сохраняются в элементы OutVal[8]...OutVal[15]. Аналогичным образом сохраняются все остальные данные. Оставшиеся биты отбрасываются.
_HIGH_LOW	Не массив	Значение бита 0 элемента In[1] сохраняется в *OutVal*.
	Массив	Значение элемента In[0] разделяется на биты, которые сохраняются в элементы OutVal[8]...OutVal[15]. Значение элемента In[1] разделяется на биты, которые сохраняются в элементы OutVal[0]...OutVal[7]. Аналогичным образом сохраняются все остальные данные. Оставшиеся биты отбрасываются.

Ниже представлен пример, в котором OutVal[] является массивом типа BOOL с 21 элементом, *Size* = UINT#3, а *Order* = _LOW_HIGH.



В показанном ниже примере OutVal[] и *Size* имеют те же значения, что и в примере выше, а *Order* = _HIGH_LOW.



Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *OutVal* является структурой, то некоторые значения In[] могут быть вставлены в дополнительные согласующие области между членами структуры в зависимости от состава структуры.
- Если значение *Size* меньше размера переменной *OutVal*, ошибки не произойдет и в *OutVal* будет сохранено указанное количество байтов данных. Если байтовых данных недостаточно, сохранятся значения, имевшиеся до выполнения команды.
  Если значение Size меньше, чем при предыдущем выполнении, предусмотрите очистку переменных с помощью команды Clear (Инициализировать) перед выполнением команды.
- Если *Size* = 0, состояние выхода *Out* поменяется на ИСТИНА, а содержимое *OutVal* не изменится.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.

- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *OutVal* не изменится.
  a) Значение *Order* находится за пределами допустимого диапазона.
  b) Значение *Size* превышает количество элементов массива In[].

SizeOfAry

Команда SizeOfAry позволяет узнать количество элементов в массиве.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SizeOfAry	Получить количество элементов массива	FUN	(@)SizeOfAry EN ENO In Out	Out:=SizeOfAry(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив	Вход	Массив	Зависит от типа данных.	---	*1
Out	Количество элементов	Выход	Количество элементов	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

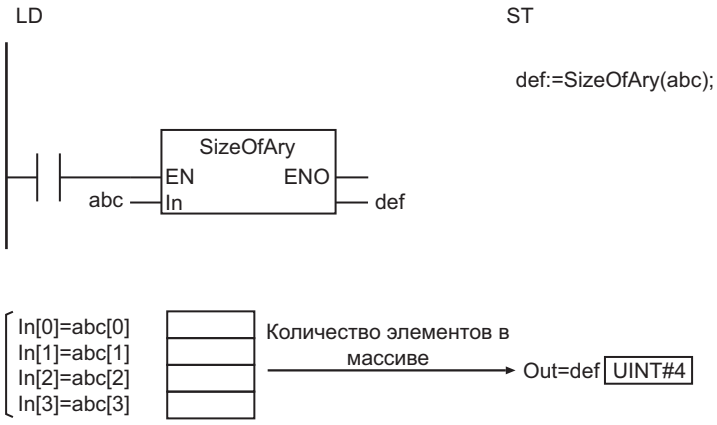
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также допускается указывать массивы перечислений или структур.																			
Out							OK													

Функция

Команда SizeOfAry возвращает количество элементов в массиве In[].
В качестве входного параметра следует использовать имя массива , например *array*, а не имя элемента массива, например array[0].

Пример программы представлен на рисунке ниже.

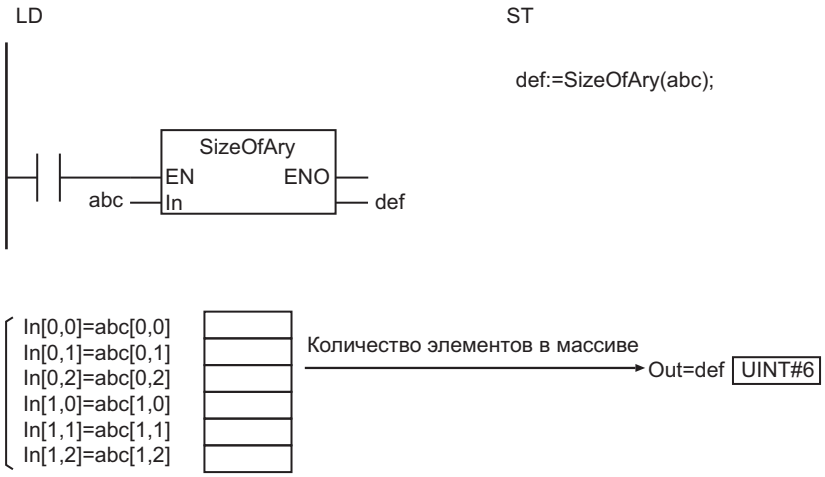
Имя	Тип данных
abc	ARRAY[0..3] OF INT



Дополнительная информация

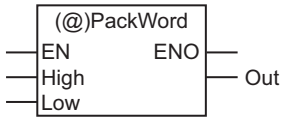
In[] может быть массивом с двумя или более измерениями. В этом случае *Out* будет содержать количество всех элементов In[].
Например, если в качестве входного параметра в In[] передается ARRAY[0..1,0..2], то *Out* будет содержать значение UINT#6.

Имя	Тип данных
abc	ARRAY[0. 1,0..2] OF BOOL



PackWord

Команда PackWord объединяет два однобайтных значения в двухбайтное значение.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
PackWord	Объединение двух байтов	FUN		Out:=PackWord(High, Low);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.12 или более поздней и Sysmac Studio версии 1.16 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
High	Старший байт данных	Вход	Данные для записи в биты 15...8 двухбайтового значения	Зависит от типа данных.	---	0
Low	Младший байт данных		Данные для записи в биты 7...0 двухбайтового значения	Зависит от типа данных.	---	0
Out	Объединенные данные	Выход	Двухбайтовое значение	Зависит от типа данных.	---	---

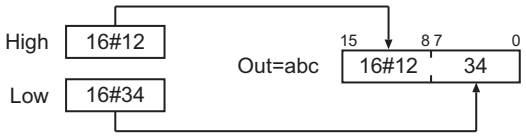
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
High		OK																		
Low		OK																		
Out			OK																	

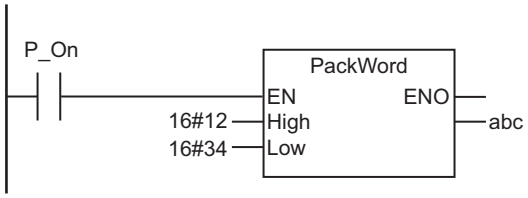
Функция

Команда PackWord объединяет два однобайтных значения в двухбайтное значение.
Значение в переменной *High* сохраняется в биты 15...8, а значение в переменной *Low* сохраняется в биты 7...0.

Программа на языке LD

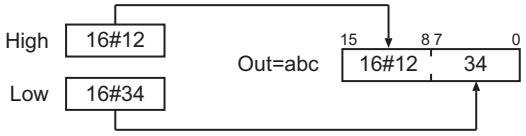
Ниже приведен пример использования команды для случая, когда *High* = 16#12, а *Low* = 16#34. Значение переменной *abc* будет равно 16#1234.





Программа на языке ST

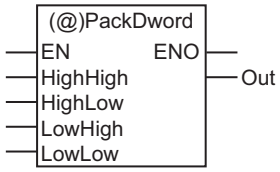
Ниже приведен пример использования команды для случая, когда *High* = 16#12, а *Low* = 16#34. Значение переменной *abc* будет равно 16#1234.




```
abc:=PackWord(16#12, 16#34);
```

PackDword

Команда PackDword объединяет четыре однобайтных значения в четырехбайтное значение.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
PackDword	Объединение четырех байтов	FUN		Out:=PackDword(HighHigh, HighLow, LowHigh, LowLow);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.12 или более поздней и Sysmac Studio версии 1.16 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
HighHigh	Старший байт старшего слова данных	Вход	Данные для записи в биты 31...24 четырехбайтового значения	Зависит от типа данных.	---	0
HighLow	Младший байт старшего слова данных		Данные для записи в биты 23...16 четырехбайтового значения	Зависит от типа данных.	---	0
LowHigh	Старший байт младшего слова данных		Данные для записи в биты 15...8 четырехбайтового значения	Зависит от типа данных.	---	0
LowLow	Младший байт младшего слова данных		Данные для записи в биты 7...0 четырехбайтового значения	Зависит от типа данных.	---	0
Out	Объединенные данные	Выход	Четырехбайтное значение	Зависит от типа данных.	---	---

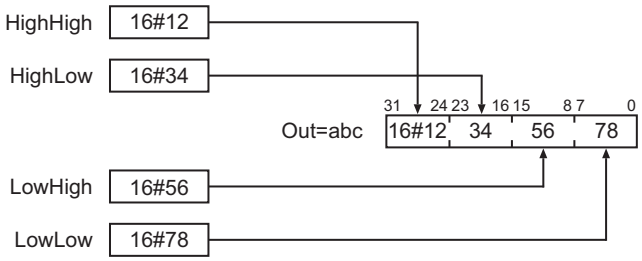
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
HighHigh		OK																		
HighLow		OK																		
LowHigh		OK																		
LowLow		OK																		
Out				OK																

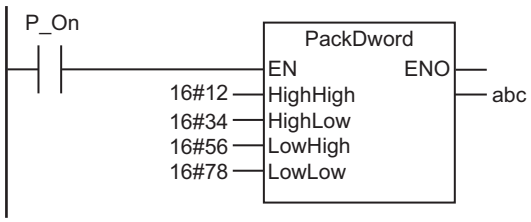
Функция

Команда PackDword объединяет четыре однобайтных значения в четырехбайтное значение. Значение, указанное в *HighHigh*, записывается в биты 31...24; значение, указанное в *HighLow*, записывается в биты 23...16; значение, указанное в *LowHigh*, записывается в биты 15...8; значение, указанное в *LowLow*, записывается в биты 7...0.

Программа на языке LD

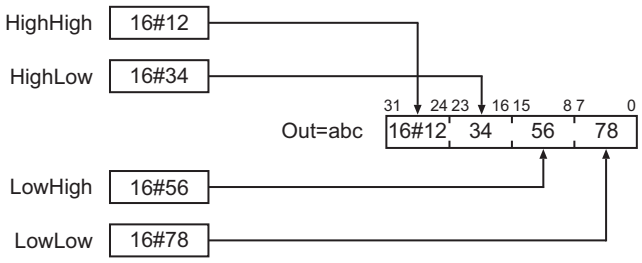
Ниже приведен пример использования команды для случая, когда *HighHigh* = 16#12, *HighLow* = 16#34, *LowHigh* = 16#56, а *LowLow* = 16#78.
Значение переменной *abc* будет равно 16#12345678.





Программа на языке ST

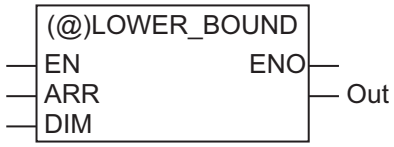
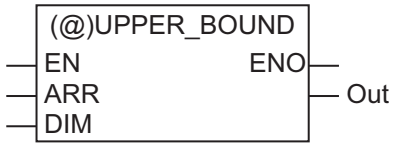
Ниже приведен пример использования команды для случая, когда *HighHigh* = 16#12, *HighLow* = 16#34, *LowHigh* = 16#56, а *LowLow* = 16#78.
Значение переменной *abc* будет равно 16#12345678.




```
abc:=PackDword(16#12, 16#34, 16#56, 16#78);
```

LOWER_BOUND и UPPER_BOUND

LOWER_BOUND : Получает минимальное значение индекса для указанного измерения массива.
UPPER_BOUND : Получает максимальное значение индекса для указанного измерения массива.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LOWER_BOUND	Получить минимальное значение индекса массива	FUN		Out:=LOWER_BOUND(ARR, DIM);
UPPER_BOUND	Получить максимальное значение индекса массива	FUN		Out:=UPPER_BOUND(ARR, DIM);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.18 или более поздней и Sysmac Studio версии 1.22 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
ARR	Массив для обработки	Вход	Массив, для которого нужно определить минимальное и максимальное значения индекса в указанном измерении. *1	---	---	---
DIM	Измерение		Указывает измерение. *2	---	---	1
Out	Возвращаемое значение	Выход	LOWER_BOUND: минимальное значение индекса UPPER_BOUND: максимальное значение индекса	Зависит от типа данных.	---	---

*1. Следует использовать имя массива, например *array*, а не имя элемента массива, например array[0].
*2. Для первого измерения массива следует указать *1*.

	Логический тип	Битовые строки				Целочисленные типы								Вещественное число		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ARR	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также допускается указывать массивы перечислений или структур.																			
DIM												OK								
Out												OK								

Функция

Команда LOWER_BOUND возвращает минимальное значение индекса для указанного в *DIM* измерения переменной-массива, указанной в *ARR*.
Аналогичным образом, команда UPPER_BOUND возвращает максимальное значение индекса для указанного в *DIM* измерения переменной-массива, указанной в *ARR*.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
P_PRGER	Флаг ошибки команды	BOOL	ИСТИНА: произошла ошибка. Будет оставаться в состоянии ИСТИНА, пока не будет сброшен в ЛОЖЬ. ЛОЖЬ: сброшен в состояние ЛОЖЬ программой пользователя.

Меры предосторожности для обеспечения надлежащей эксплуатации

В указанных ниже случаях происходит ошибка. Выход *ENO* перейдет в состояние ЛОЖЬ, а содержимое *Out* не изменится.

- *ARR* не является массивом.
- Указанное в *DIM* значение меньше или равно *0* либо превышает число измерений, имеющихся в *ARR*.

Пример программы

Вычисление суммы элементов массива

В этом примере программы будет показано, как определить одномерную переменную-массив переменной длины и как получить минимальное и максимальное значения индекса переменной-массива переменной длины.

● Программа пользовательской функции (Sum)

Внутренняя переменная	Имя	Тип данных	По умолчанию	Комментарий
	i	DINT		

Входные/выходные переменные	Имя	Вход/выход	Тип данных	Комментарий
	EN	Вход	BOOL	
	ENO	Выход	BOOL	
	a	Вход-выход	ARRAY[*] OF INT	

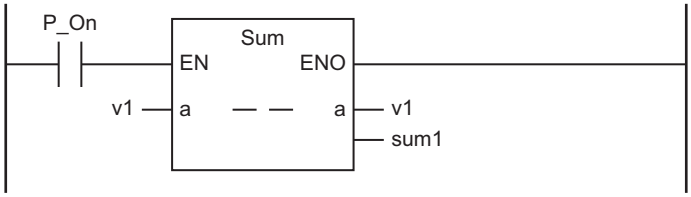
Возвращаемое значение	Имя	Тип данных	По умолчанию	Комментарий
	Sum	INT		

```
Sum := 0;
FOR i := LOWER_BOUND(a,1) TO UPPER_BOUND(a,1) DO
    Sum := Sum + a[i];
END_FOR;
```

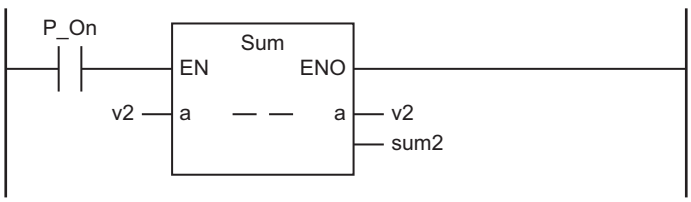
● Вызывающая программа

Внутренние переменные	Имя	Тип данных	По умолчанию	Комментарий
	v1	ARRAY[0..4] OF INT	[1,2,3,4,5]	
	v2	ARRAY[0..9] OF INT	[1,2,3,4,5,6,7,8,9,10]	
	sum1	INT		
	sum2	INT		

Sum1 = 1+2+3+4+5 =15



Sum2 = 1+2+3+4+5+6+7+8+9+10 =55



Сложение матриц 2× 2

В этом примере программы будет показано, как определить многомерную переменную-массив переменной длины и как использовать команды LOWER_BOUND и UPPER_BOUND для многомерной переменной-массива переменной длины.

● Программа пользовательской функции (Matrix_Add)

Внутренние переменные	Имя	Тип данных	По умолчанию	Комментарий
	i	DINT		
	j	DINT		
	m1	DINT		
	m2	DINT		
	n1	DINT		
	n2	DINT		

Входные/выходные переменные	Имя	Вход/выход	Тип данных	Комментарий
	EN	Вход	BOOL	
	ENO	Выход	BOOL	
	A	Вход-выход	ARRAY[*,*] OF DINT	
	B	Вход-выход	ARRAY[*,*] OF DINT	
	C	Вход-выход	ARRAY[*,*] OF DINT	

Возвращаемое значение	Имя	Тип данных	По умолчанию	Комментарий
	Matrix_Add	BOOL		

```
m1 := LOWER_BOUND(C,1);
m2 := UPPER_BOUND(C,1);
n1 := LOWER_BOUND(C,2);
n2 := UPPER_BOUND(C,2);

FOR i := m1 TO m2 DO
    FOR j := n1 TO n2 DO
        C[i,j] := A[i,j] + B[i,j];
    END_FOR;
END_FOR;
```

● Вызывающая программа

Внутренние переменные	Имя	Тип данных	По умолчанию	Комментарий
	X	ARRAY[0..1,0..1] OF DINT	[0, 1, 2, 3]	
	Y	ARRAY[0..1,0..1] OF DINT	[1, 2, 3, 4]	
	Z	ARRAY[0..1,0..1] OF DINT		

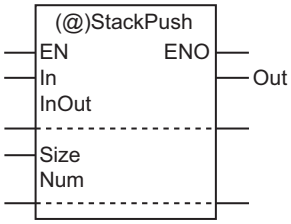
```
// Z = X + Y = |0 1| + |1 2| = |1 3|
//             |2 3|   |3 4|   |5 7|
Matrix_Add(X, Y, Z);
```

Команды для работы со стеком и таблицами

Команда	Имя	Стр.
StackPush	Ввод в стек	стр. 2-566
StackFIFO и StackLIFO	Первым вошел — первым вышел/Последним вошел — первым вышел	стр. 2-575
StackIns	Вставка в стек	стр. 2-579
StackDel	Удаление из стека	стр. 2-582
RecSearch	Поиск записи	стр. 2-584
RecRangeSearch	Поиск диапазона записей	стр. 2-589
RecSort	Сортировка записей	стр. 2-594
RecNum	Получить количество записей	стр. 2-601
RecMax и RecMin	Поиск записи с максимальным значением/Поиск записи с минимальным значением	стр. 2-604

StackPush

Команда StackPush сохраняет значение на вершину стека.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
StackPush	Ввод в стек	FUN		StackPush(In, InOut, Size, Num);

Переменные

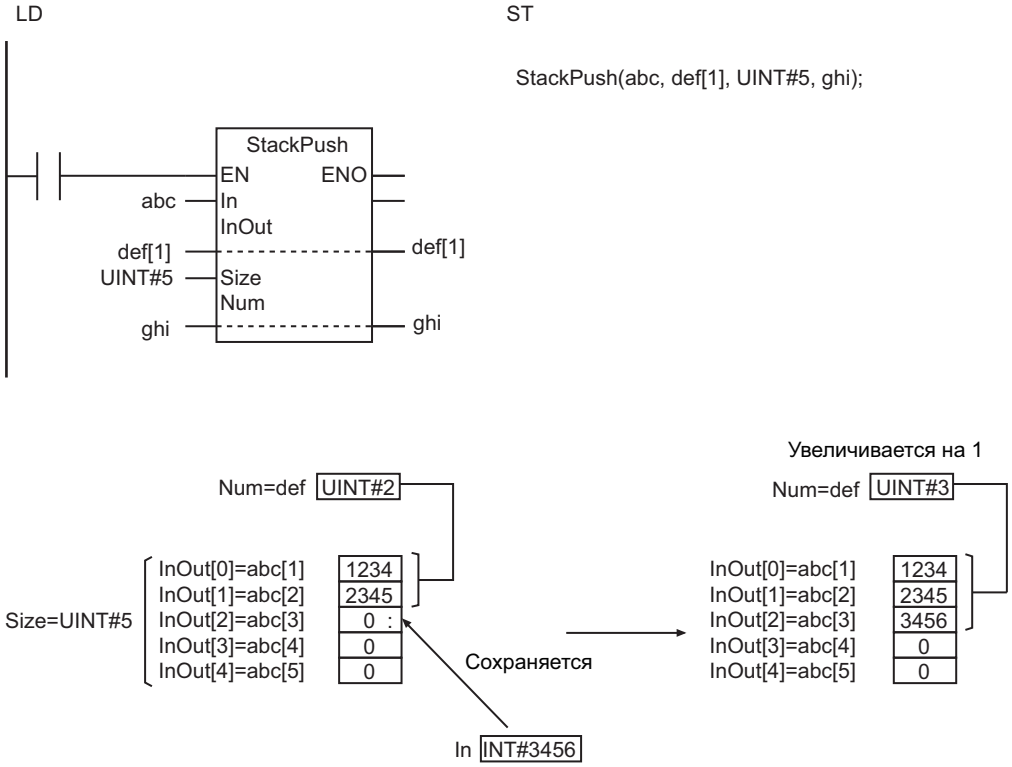
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Входное значение	Вход	Значение, структура или член структуры для размещения в стеке	Зависит от типа данных.	---	---
Size	Количество элементов стека		Количество элементов в массиве стека			1
InOut[] (массив)	Массив стека	Вход-выход	Массив, выполняющий функцию стека	Зависит от типа данных.	---	---
Num	Количество сохраненных элементов		Количество элементов, сохраненных в стек			
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также можно указать перечисление, структуру или член структуры.																			
Size							ОК													
InOut[] (массив)	Должен представлять собой массив с элементами того же типа данных, что и значение *In*.																			
Num							ОК													
Out	ОК																			

Функция

Команда StackPush предполагает, что в массиве стека InOut[] хранится *Num* элементов, и записывает значение *In* (входное значение) в следующий элемент стека, т. е. в InOut[*Num*].
После этого значение *Num* увеличивается на 1.
В параметре *Size* (количество элементов стека) нужно указать количество элементов массива InOut[], которые используются для стека.

Ниже показан пример для случая, когда *Size* = UINT#5, а *Num* = UINT#2.



Дополнительная информация

Для удаления самого нижнего или самого верхнего значения, хранящегося в стеке, используйте команду *StackFIFO u StackLIFO* на стр. 2-575.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте одинаковый тип данных для переменной *In* и элементов InOut[]. При использовании разных типов данных произойдет ошибка сборки.
- Если в InOut[] будет передан элемент массива, будут обработаны все элементы, расположенные ниже переданного элемента.
- Если значение *Size* равно 0, то значения InOut[] и *Num* не изменяются.
- Всегда используйте переменную в качестве входного параметра, передаваемого в *In*. При передаче константы произойдет ошибка сборки.
- Если параметр *In* является перечислением, передать непосредственно перечислитель невозможно. В случае непосредственной передачи перечислителя произойдет ошибка сборки.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.

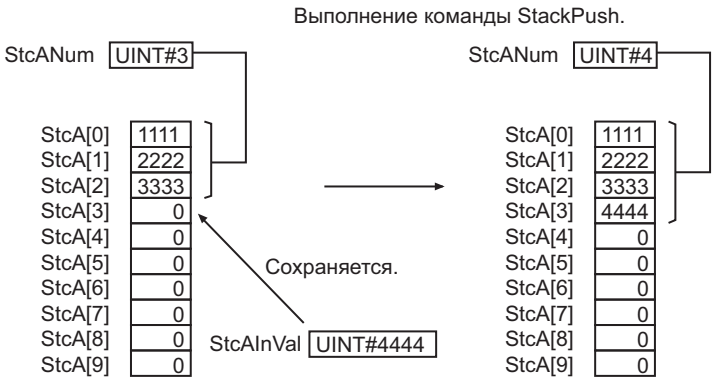
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое InOut[] не изменится.
  a) Значение *Size* не равно 0, при этом *Num* больше или равно *Size*.
  b) Значение *Size* приводит к выходу за область массива InOut[].
  c) *In* и InOut[] содержат строковые данные (STRING), при этом количество байтов в *In* превышает размер InOut[].

Пример программы

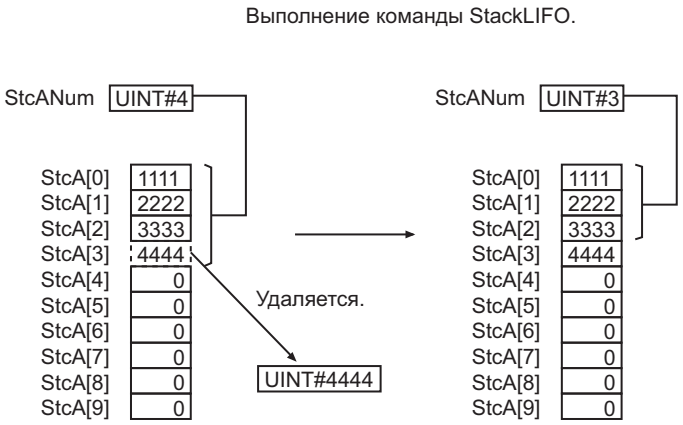
В качестве стека используется переменная-массив StcA[0..9]. Предварительно в стек были сохранены три значения: UINT#1111, UINT#2222 и UINT#3333.

StcA[0]	1111
StcA[1]	2222
StcA[2]	3333
StcA[3]	0
StcA[4]	0
StcA[5]	0
StcA[6]	0
StcA[7]	0
StcA[8]	0
StcA[9]	0

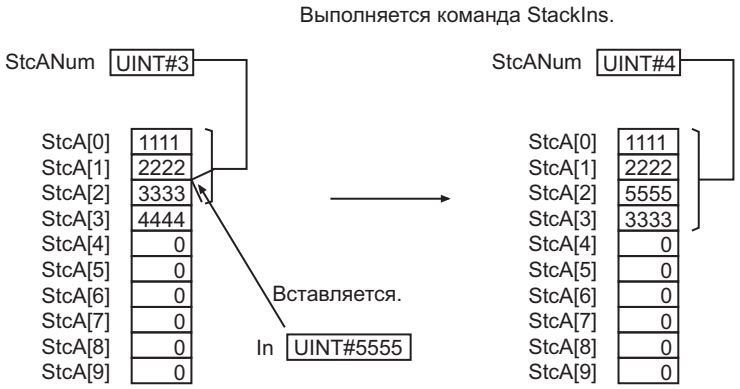
Используется команда StackPush для размещения нового значения (UINT#4444) в вершину стека, т. е. в StcA[3]. Это означает, что в стеке будет четыре значения.



После этого используется команда StackLIFO для удаления самого верхнего значения стека, т. е. StcA[3]. Это означает, что в стеке будет три значения.

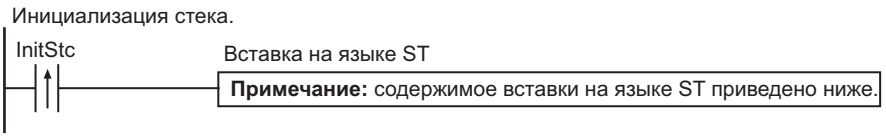


Наконец, используется команда StackIns для вставки значения (UINT#5555) между элементами StcA[1] и StcA[2]. Это означает, что в стеке будет четыре значения.

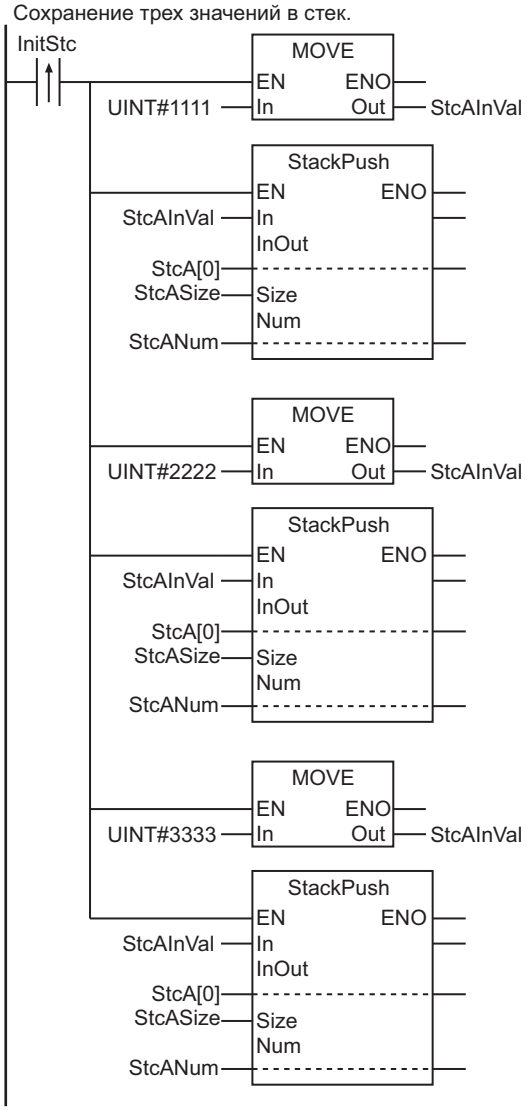


Программа на языке LD

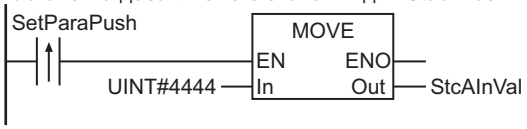
Переменная	Тип данных	Начальное значение	Комментарий
InitStc	BOOL	ЛОЖЬ	Условие инициализации стека
StcANum	UINT	0	Количество сохраненных элементов
StcA	ARRAY[0..9] OF UINT	[10(0)]	Массив стека
StcASize	UINT	0	Количество элементов стека
SetParaPush	BOOL	ЛОЖЬ	Условие выполнения для установки StcAInVal.
StcAInVal	UINT	0	Значение, добавляемое командой StackPush
StcAPushStat	BOOL	ЛОЖЬ	Условие выполнения StackPush
StackPush_err	BOOL	ЛОЖЬ	Флаг ошибки StackPush
StcALIFOStat	BOOL	ЛОЖЬ	Условие выполнения StackLIFO
StcAOutVal	UINT	0	Значение, удаляемое командой StackLIFO
StackLIFO_err	BOOL	ЛОЖЬ	Флаги ошибки StackLIFO
SetParaIns	BOOL	ЛОЖЬ	Условие выполнения для установки StcAInsVal и StcAOffset
StcAInsVal	UINT	0	Значение, вставляемое командой StackIns
StcAOffset	UINT	0	Смещение для StackIns
StcAInsStat	BOOL	ЛОЖЬ	Условие выполнения StackIns
StackIns_err	BOOL	ЛОЖЬ	Флаги ошибки StackIns



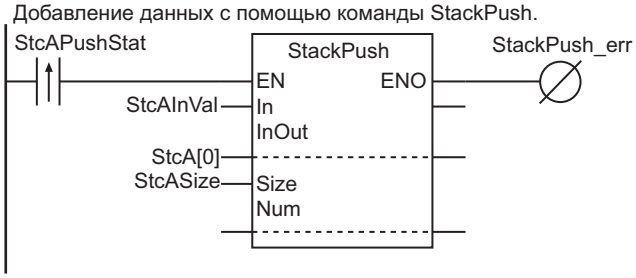
Сохранение трех значений в стек.



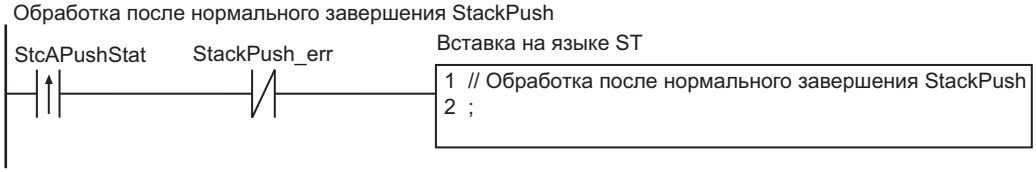
Установка добавляемого значения для StackPush.



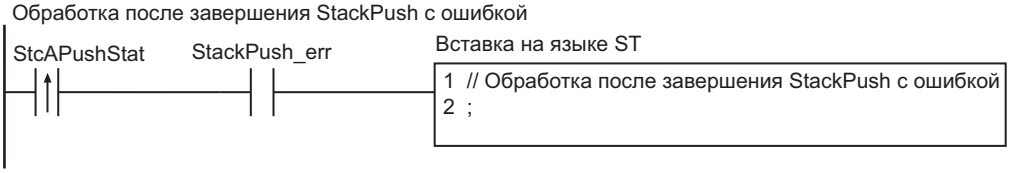
Добавление данных с помощью команды StackPush.



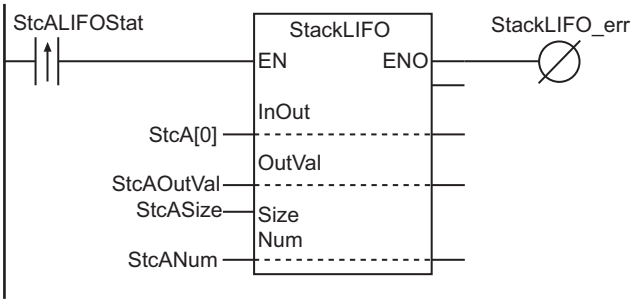
Обработка после нормального завершения StackPush



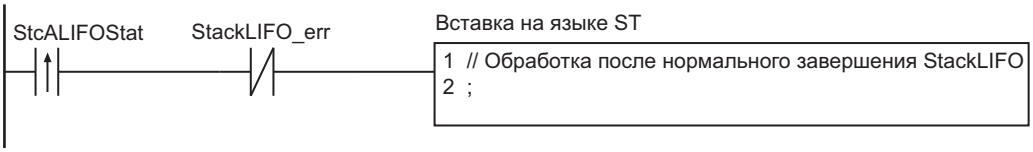
Обработка после завершения StackPush с ошибкой



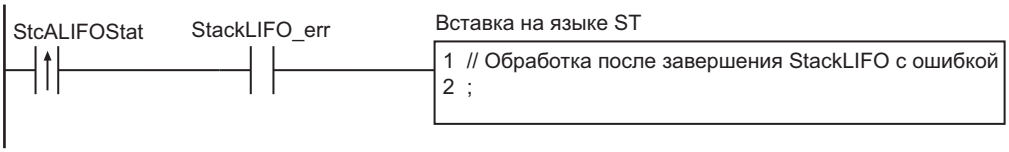
Удаление данных с помощью команды StackLIFO.



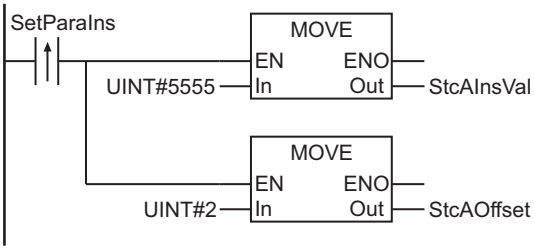
Обработка после нормального завершения StackLIFO



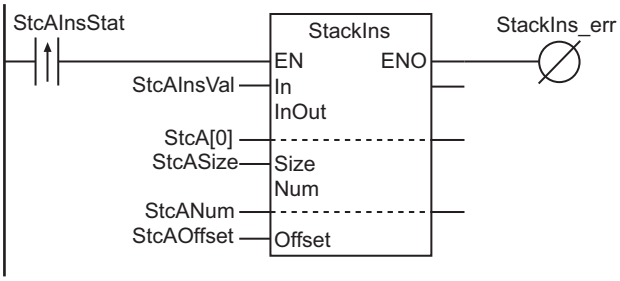
Обработка после завершения StackLIFO с ошибкой



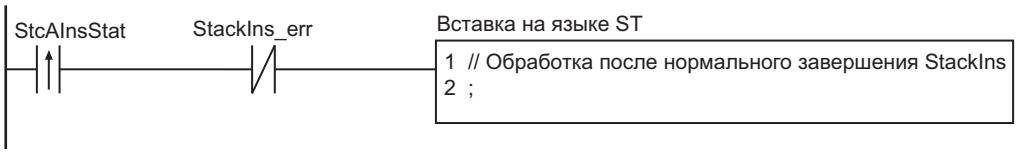
Установка вставляемого значения и смещения для команды StackInsh.



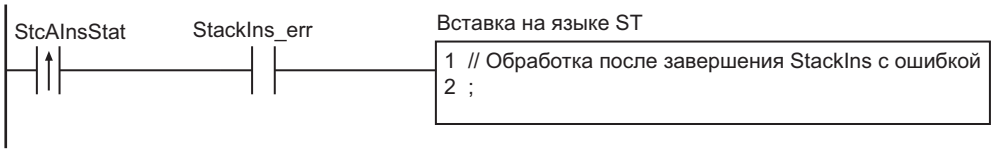
Вставка данных с помощью команды StackIns.



Обработка после нормального завершения StackIns



Обработка после завершения StackIns с ошибкой



● Содержимое вставки на языке ST

```
StcANum:=0;
Clear(StcA);
StcASize:=SizeOfAry(StcA);
```

Программа на языке ST

Переменная	Тип данных	Начальное значение	Комментарий
InitStc	BOOL	ЛОЖЬ	Условие инициализации стека
preInitStc	BOOL	ЛОЖЬ	Значение InitStc в предыдущем цикле выполнения задачи
StcANum	UINT	0	Количество сохраненных элементов
StcA	ARRAY[0..9] OF UINT	[10(0)]	Массив стека
StcASize	UINT	0	Количество элементов стека
StcAPushStat	BOOL	ЛОЖЬ	Условие выполнения StackPush
preStcAPushStat	BOOL	ЛОЖЬ	Значение StcAPushStat в предыдущем цикле выполнения задачи
StcAInVal	UINT	0	Значение, добавляемое командой StackPush
StcAPush_OK	BOOL	ЛОЖЬ	Флаг нормального завершения StackPush
StcAPushNormalEnd	BOOL	ЛОЖЬ	Обработка после нормального завершения StackPush
StcAPushErrorEnd	BOOL	ЛОЖЬ	Обработка после завершения StackPush с ошибкой
StcALIFOStat	BOOL	ЛОЖЬ	Условие выполнения StackLIFO
preStcALIFOStat	BOOL	ЛОЖЬ	Значение StcALIFOStat в предыдущем цикле выполнения задачи
StcAOutVal	UINT	0	Значение, удаляемое командой StackLIFO
StcALIFO_OK	BOOL	ЛОЖЬ	Флаг нормального завершения StackLIFO
StcALIFONormalEnd	BOOL	ЛОЖЬ	Обработка после нормального завершения StackLIFO
StcALIFOErrorEnd	BOOL	ЛОЖЬ	Обработка после завершения StackLIFO с ошибкой
StcAInsStat	BOOL	ЛОЖЬ	Условие выполнения StackIns
preStcAInsStat	BOOL	ЛОЖЬ	Значение StcAInsStat в предыдущем цикле выполнения задачи
StcAInsVal	UINT	0	Значение, вставляемое командой StackIns
StcAOffset	UINT	0	Смещение для StackIns

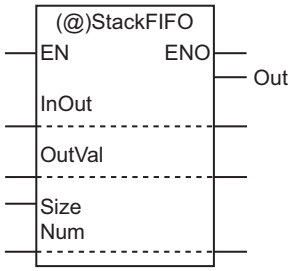
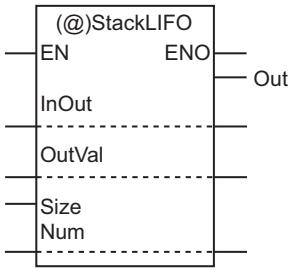
Переменная	Тип данных	Начальное значение	Комментарий
StcAIns_OK	BOOL	ЛОЖЬ	Флаг нормального завершения StackIns
StcAInsNormalEnd	BOOL	ЛОЖЬ	Обработка после нормального завершения StackIns
StcAInsErrorEnd	BOOL	ЛОЖЬ	Обработка после завершения StackIns с ошибкой

```
// Инициализировать стек.
IF ( (InitStc=TRUE) AND (preInitStc=FALSE) ) THEN
  StcANum:=0;
  Clear(StcA);
  StcASize:=SizeOfAry(StcA);
END_IF;

// Сохранить три значения в стек.
IF ( (InitStc=TRUE) AND (preInitStc=FALSE) ) THEN
  StackPush(In:=UINT#1111, InOut:=StcA[0], Size:=StcASize, Num:=StcANum);
  StackPush(In:=UINT#2222, InOut:=StcA[0], Size:=StcASize, Num:=StcANum);
  StackPush(In:=UINT#3333, InOut:=StcA[0], Size:=StcASize, Num:=StcANum);
END_IF;

preInitStc:=InitStc;

// Добавить данные с помощью команды StackPush.
IF ( (StcAPushStat=TRUE) AND (preStcAPushStat=FALSE) ) THEN
  StcAInVal:=UINT#4444;
  StackPush(
    In :=StcAInVal, // Добавляемое значение
    InOut:=StcA[0], // Первый элемент в массиве стека
    Size :=StcASize, // Количество элементов стека
    Num :=StcANum, // Количество сохраненных элементов
    ENO =>StcAPush_OK); // Флаг нормального завершения
  IF (StcAPush_OK=TRUE) THEN
    StcAPushNormalEnd:=TRUE;// Обработка после нормального завершения
  ELSE
    StcAPushErrorEnd:=TRUE; // Обработка после завершения с ошибкой
  END_IF;
END_IF;

preStcAPushStat:=StcAPushStat;
// Удалить данные с помощью команды StackLIFO.
IF ( (StcALIFOStat=TRUE) AND (preStcALIFOStat=FALSE) ) THEN
  StackLIFO(
    InOut :=StcA[0], // Первый элемент в массиве стека
    OutVal :=StcAOutVal, // Значение, удаляемое из стека
    Size :=StcASize, // Количество элементов стека
    Num :=StcANum, // Количество сохраненных элементов
    ENO =>StcALIFO_OK); // Флаг нормального завершения
  IF (StcALIFO_OK=TRUE) THEN
    StcALIFONormalEnd:=TRUE; // Обработка после нормального завершения
```

```
  ELSE
    StcALIFOErrorEnd:=TRUE; // Обработка после завершения с ошибкой
  END_IF;
END_IF;
preStcALIFOStat:=StcALIFOStat;

// Вставить данные с помощью команды StackIns.
IF ( (StcAInsStat=TRUE) AND (preStcAInsStat=FALSE) ) THEN
  StcAInsVal:=UINT#5555;
  StcAOffset:=UINT#2;
  StackIns(
    In :=StcAInsVal, // Значение, вставляемое в стек
    InOut :=StcA[0], // Первый элемент в массиве стека
    Size :=StcASize, // Количество элементов стека
    Num :=StcANum, // Количество сохраненных элементов
    Offset:=StcAOffset, // Смещение для вставки значения
    ENO =>StcAIns_OK); // Флаг нормального завершения
  IF (StcAIns_OK=TRUE) THEN
    StcAInsNormalEnd:=TRUE; // Обработка после нормального завершения
  ELSE
    StcAInsErrorEnd:=TRUE; // Обработка после завершения с ошибкой
  END_IF;
END_IF;
preStcAInsStat:=StcAInsStat;
```

StackFIFO и StackLIFO

StackFIFO : Удаляет самое нижнее значение стека.
StackLIFO : Удаляет самое верхнее значение стека.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
StackFIFO	Первым вошел — первым вышел	FUN		StackFIFO(InOut, OutVal, Size, Num);
StackLIFO	Последним вошел — первым вышел	FUN		StackLIFO(InOut, OutVal, Size, Num);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Size	Количество элементов стека	Вход	Количество элементов в массиве стека	Зависит от типа данных.	---	1
InOut[] (массив)	Массив стека	Вход-выход	Массив, выполняющий функцию стека	Зависит от типа данных.	---	---
OutVal	Выходное значение		Значение или структура, выведенные из стека			
Num	Количество сохраненных элементов		Количество элементов, сохраненных в стек			
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size							ОК													
InOut[] (массив)	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также допускается указывать массивы перечислений или структур.																			
OutVal	Тип данных должен быть таким же, как у элементов массива InOut[].																			
Num							ОК													
Out	ОК																			

Функция

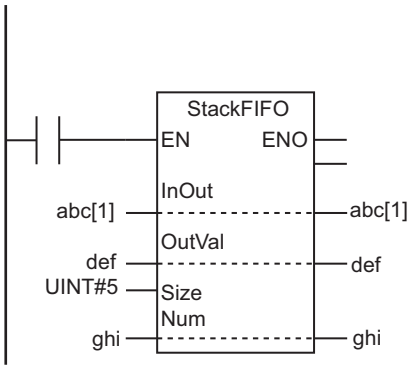
Команда предполагает, что в массиве стека InOut[] хранится *Num* элементов. Команда удаляет значение из стека и присваивает его переменной *OutVal* (выходное значение).
В параметре *Size* (количество элементов стека) нужно указать количество элементов массива InOut[], которые используются как стек.

StackFIFO

Команда StackFIFO извлекает значение, хранящееся на дне стека. Значение InOut[0] присваивается переменной *OutVal*.
Затем все остальные *Num*-1 элементов, начиная с InOut[1], сдвигаются на одну позицию вниз (т. е. значение каждого элемента массива стека передается в элемент с индексом на 1 меньше). После этого значение *Num* уменьшается на 1.

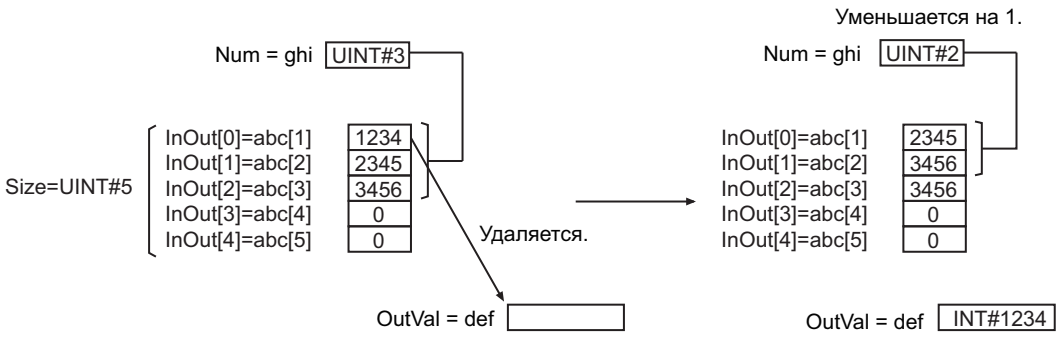
Ниже показан пример для случая, когда *Size* = UINT#5, а *Num* = UINT#3.

LD



ST

```
StackFIFO(abc[1], def, UINT#5, ghi);
```

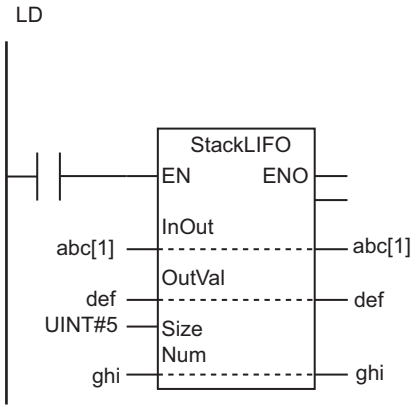


StackLIFO

Команда StackLIFO извлекает значение, хранящееся на вершине стека. Значение InOut[*Num*-1] присваивается переменной *OutVal*.

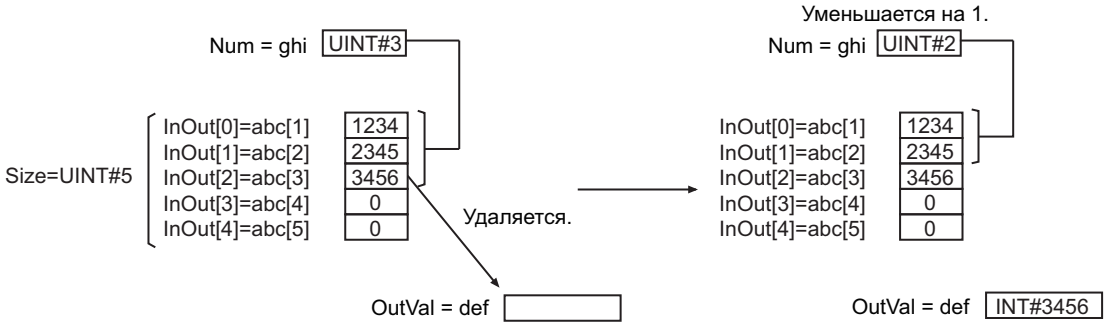
После этого значение *Num* уменьшается на 1.

Ниже показан пример для случая, когда *Size* = UINT#5, а *Num* = UINT#2.



ST

```
StackLIFO(abc[1], def, UINT#5, ghi);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

- Для InOut[] и *OutVal* следует использовать один и тот же тип данных. При использовании разных типов данных произойдет ошибка сборки.
- Если в InOut[] будет передан элемент массива, будут обработаны все элементы, расположенные ниже переданного элемента.
- Если значение *Size* или *Num* равно 0, то значения в InOut[], *Num* и *OutVal* не изменяются.

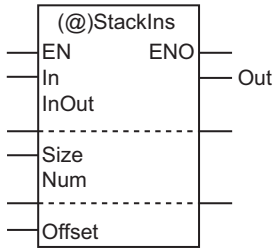
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *OutVal* не изменится.
  a) Значения *Num* и *Size* не равны 0, при этом *Num* больше *Size*.
  b) Значение *Size* приводит к выходу за область массива InOut[].
  c) InOut[] является строковым массивом (STRING), и какой-либо из его элементов не завершается символом NULL.
  d) InOut[] является строковым массивом (STRING), и общее количество байтов в элементах превышает размер *OutVal*.

Пример программы

См. *Пример программы* на стр. 2-568 для команды StackPush.

StackIns

Команда StackIns вставляет значение в указанную позицию стека.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
StackIns	Вставка в стек	FUN		StackIns(In, InOut, Size, Num, Offset);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Вставляемое значение	Вход	Значение, структура или член структуры для вставки в стек	Зависит от типа данных.	---	*1
Size	Количество элементов стека		Количество элементов в массиве стека			1
Offset	Смещение		Позиция в стеке для вставки *In*			0
InOut[] (массив)	Массив стека	Вход-выход	Массив, выполняющий функцию стека	Зависит от типа данных.	---	---
Num	Количество сохраненных элементов		Количество элементов, сохраненных в стек			
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать перечисление, структуру или член структуры.																			
Size							OK													
Offset							OK													
InOut[] (массив)	Должен представлять собой массив с элементами того же типа данных, что и значение *In*.																			
Num							OK													

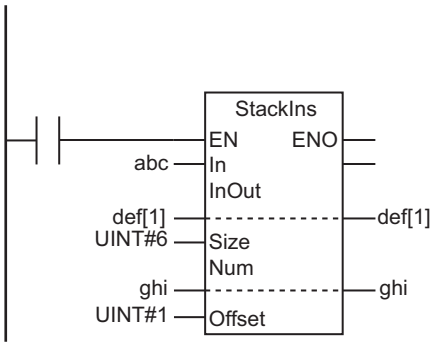
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	OK																			

Функция

Команда StackIns предполагает, что в массиве стека InOut[] хранится *Num* элементов, и вставляет значение *In* (вставляемое значение) в позицию элемента InOut[*Offset*], заданную параметром *Offset*.
Все элементы, расположенные выше этой позиции, т. е. InOut[*Offset*]...InOut[*Num*-1], сдвигаются на одну позицию вверх (т. е. значение каждого элемента массива стека передается в элемент с индексом на 1 больше).
После этого значение *Num* увеличивается на 1.
В параметре *Size* (количество элементов стека) нужно указать количество элементов массива InOut[], которые используются для стека.

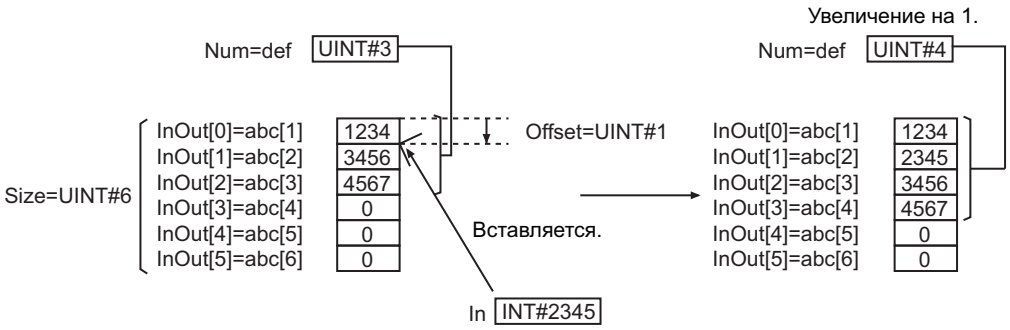
Ниже показан пример для случая, когда *Size* = UINT#6, *Num* = UINT#3, а *Offset* = UINT#1.

LD



ST

```
StackIns(abc, def[1], UINT#6, ghi, UINT#1);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте одинаковый тип данных для переменных *In* и InOut[]. При использовании разных типов данных произойдет ошибка сборки.

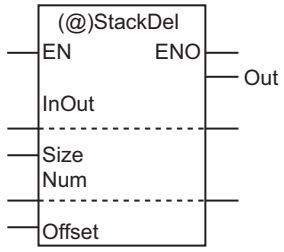
- Если в InOut[] будет передан элемент массива, будут обработаны все элементы, расположенные ниже переданного элемента.
- Если значение *Size* равно 0, то значения InOut[] и *Num* не изменяются.
- Всегда используйте переменную в качестве входного параметра, передаваемого в *In*. При передаче константы произойдет ошибка сборки.
- Если параметр *In* является перечислением, передать непосредственно перечислитель невозможно. В случае непосредственной передачи перечислителя произойдет ошибка сборки.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое InOut[] не изменится.
  a) Значение *Size* не равно 0, при этом значение *Size* не больше значения *Num*, которое не больше или равно значению *Offset*.
  b) Значение *Size* приводит к выходу за область массива InOut[].
  c) *In* и InOut[] содержат строковые данные (STRING), при этом количество байтов в *In* превышает размер InOut[].

Пример программы

См. *Пример программы* на стр. 2-568 для команды StackPush.

StackDel

Команда StackDel удаляет значение из указанной позиции в стеке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
StackDel	Удаление из стека	FUN		StackDel(InOut, Size, Num, Offset);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Size	Количество элементов стека	Вход	Количество элементов в массиве стека	Зависит от типа данных.	---	1
Offset	Смещение		Позиция значения, удаляемого из стека			0
InOut[] (массив)	Массив стека	Вход-выход	Массив, выполняющий функцию стека	Зависит от типа данных.	---	---
Num	Количество сохраненных элементов		Количество элементов, сохраненных в стек			
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

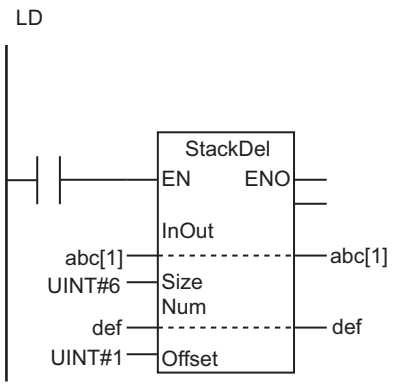
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size							ОК													
Offset							ОК													
InOut[] (массив)	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также допускается указывать массивы перечислений или структур.																			
Num							ОК													
Out	ОК																			

Функция

Команда StackDel предполагает, что в массиве стека InOut[] хранится *Num* элементов, и удаляет значение в позиции элемента InOut[*Offset*], заданной параметром *Offset*.

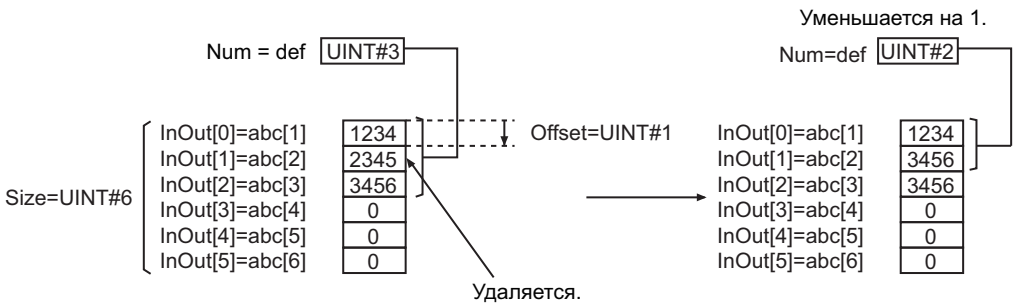
Все элементы, расположенные выше этой позиции, т. е. InOut[*Offset*+1]...InOut[*Num*-1], сдвигаются на одну позицию вниз (т. е. значение каждого элемента массива стека передается в элемент с индексом на 1 меньше).
После этого значение *Num* уменьшается на 1.
В параметре *Size* (количество элементов стека) нужно указать количество элементов массива InOut[], которые используются для стека.

Ниже показан пример для случая, когда *Size* = UINT#6, *Num* = UINT#3, а *Offset* = UINT#1.



ST

```
StackDel(abc[1], UINT#6, def, UINT#1);
```

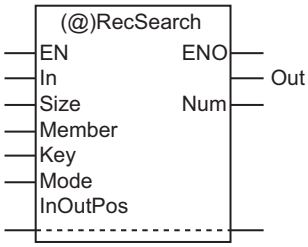


Меры предосторожности для обеспечения надлежащей эксплуатации

- Если в InOut[] будет передан элемент массива, будут обработаны все элементы, расположенные ниже переданного элемента.
- Если значение *Size* или *Num* равно 0, то значения InOut[] и *Num* не изменяются.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое InOut[] не изменится.
  a) Значения *Num* и *Size* не равны 0, при этом значение *Size* не больше или равно значению *Num*, которое не больше значения *Offset*.
  b) Значение *Size* приводит к выходу за область массива InOut[].

RecSearch

Команда RecSearch производит поиск элементов, содержащих заданное значение, среди элементов массива структур, используя указанный метод поиска.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
RecSearch	Поиск записи	FUN		Out:=RecSearch(In, Size, Member, Key, Mode, InOutPos, Num);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для поиска	Вход	Массив структур для поиска	---	---	*1
Size	Количество элементов для поиска		Количество элементов массива для поиска	Зависит от типа данных.	---	1
Member	Член структуры для поиска		Член структуры In[] для поиска			*1
Key	Ключ поиска		Искомое значение			
Mode	Метод поиска		Метод поиска	_LINEAR, _BIN_ASC, _BIN_DESC		_LINEAR
InOutPos[] (массив)	Номера элементов с совпадающим значением	Вход-выход	Номера элементов, удовлетворяющих условиям поиска	Зависит от типа данных.	---	---
Out	Результат поиска	Выход	ИСТИНА: есть элементы, соответствующие условиям поиска ЛОЖЬ: нет элементов, соответствующих условиям поиска	Зависит от типа данных.	---	---
Num	Число совпадений		Число совпадений			

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)	Следует указать массив структур.																			
Size							ОК													
Member						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	OK *1	OK *1	OK *1	OK *1	OK *1
	Следует указать такой же тип данных, как у члена структуры для поиска In[].																			
Key	Тип данных должен быть таким же, как у *Member*.																			
Mode	Сведения о перечислителях перечислимого типа _eSEARCH_MODE см. в разделе *Функция* на стр. 2-585.																			
InOutPos[] (массив)							ОК													
Out	ОК																			
Num							ОК													

*1. При использовании модуля ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше можно указать следующий тип: TIME, DATE, TOD, DT и STRING.

Функция

Команда RecSearch производит поиск среди *Size* элементов массива структур In[], т. е. среди элементов с In[0] по In[*Size*-1]. Ищутся элементы, в которых содержимое члена структуры *Member* (член структуры для поиска) совпадает с искомым значением *Key* (ключ поиска).
Член структуры элемента массива In[], в котором должен производиться поиск, передается в качестве аргумента в параметр *Member*.
При обнаружении хотя бы одного элемента, отвечающего условиям поиска, значение переменной *Out* (результат поиска) меняется на ИСТИНА. Номер найденного элемента присваивается элементу массива InOutPos[0], а в переменную *Num* записывается количество найденных элементов. Если найдено несколько элементов с искомым значением, элементу InOutPos[0] присваивается наименьший номер найденного элемента массива In[].
Если ни одного элемента не будет найдено, выход *Out* будет находиться в состоянии ЛОЖЬ, а InOutPos[0] и *Num* будут равны 0.

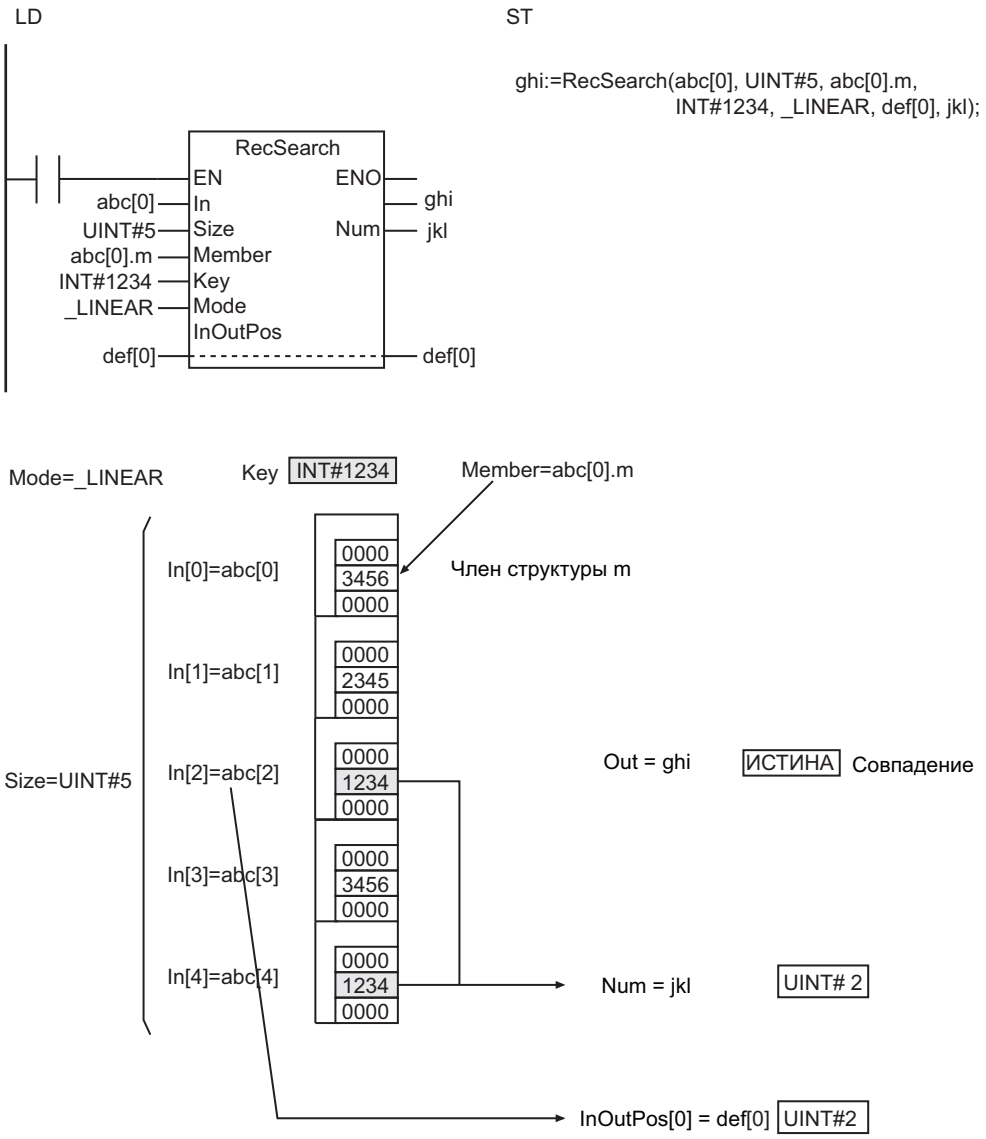
Всегда присоединяйте номер элемента к входному параметру, который передается в In[], например: array[3].

Для параметра *Mode* используется перечислимый тип данных _eSEARCH_MODE. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_LINEAR	Линейный поиск
_BIN_ASC	Бинарный поиск в порядке возрастания
_BIN_DESC	Бинарный поиск в порядке убывания

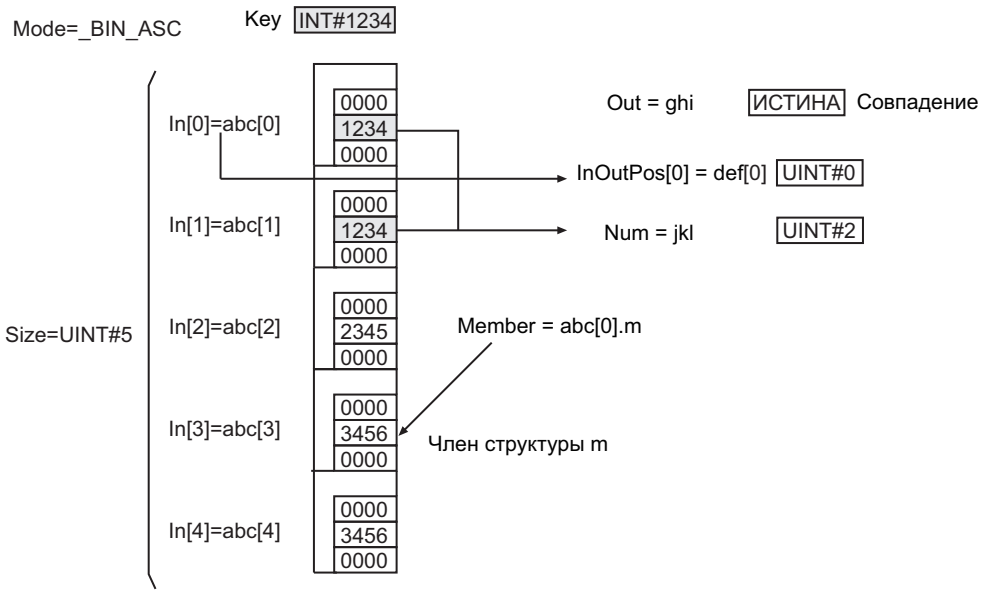
При линейном поиске поиск выполняется по порядку, начиная с первого элемента массива In[].

Ниже показан пример для случая, когда *Size* = UINT#5, *Key* = INT#1234 и *Mode* = _LINEAR.

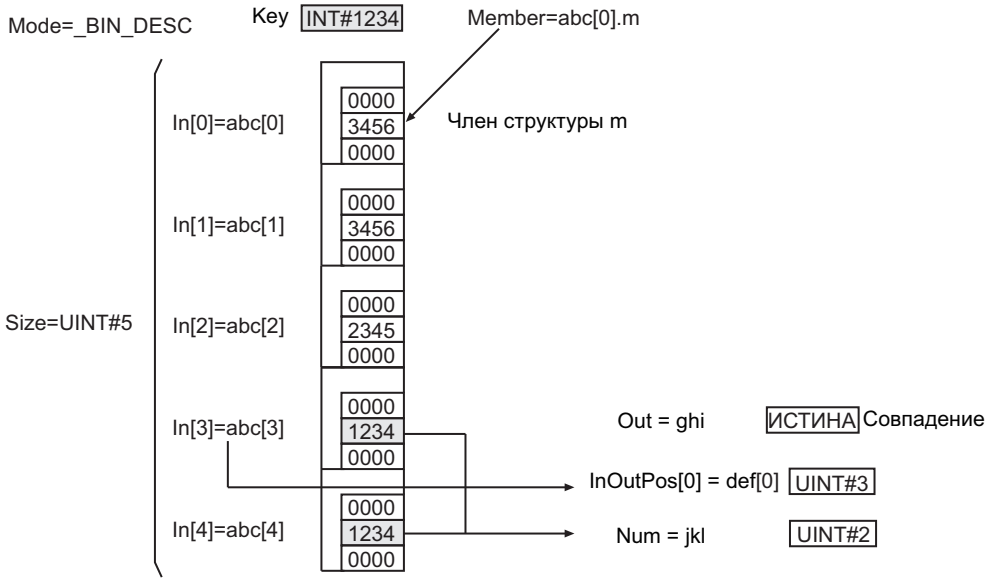


При бинарном поиске в порядке возрастания элементы массива, передаваемого в качестве входного параметра в переменную In[], должны быть заранее отсортированы в порядке возрастания. Затем выполняется команда для выполнения бинарного поиска.

В следующем примере используются те же исходные данные, что в примере выше, однако элементы массива отсортированы в порядке возрастания. Производится бинарный поиск, результаты которого показаны ниже.



При бинарном поиске в порядке убывания элементы массива, передаваемого в качестве входного параметра в переменную In[], должны быть заранее отсортированы в порядке убывания. Затем выполняется команда для выполнения бинарного поиска.
В следующем примере используются те же исходные данные, что и в предыдущем примере, однако элементы массива отсортированы в порядке убывания. Производится бинарный поиск, результаты которого показаны ниже.



Дополнительная информация

- Массив In[] может быть членом структуры более высокого уровня.
  Пример: In[0]=str0.str1[0]
- In[] может быть массивом с двумя или более измерениями. Если In[] является двумерным массивом, номер соответствующего условиям поиска элемента, найденного в первом измерении, присваивается элементу InOutPos[0], а номер элемента, найденного во втором измерении, присваивается элементу InOutPos[1].

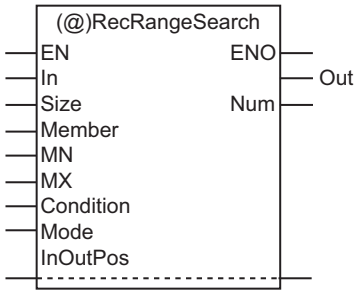
- Если In[] является трехмерным массивом, номер соответствующего условиям поиска элемента, найденного в первом измерении, присваивается элементу InOutPos[0]; номер элемента, найденного во втором измерении, присваивается элементу InOutPos[1]; номер элемента, найденного в третьем измерении, присваивается элементу InOutPos[2].
- При поиске значения типа TIME, DT или TOD необходимо установить одинаковую точность для значений параметров *Member* и *Key*. Для установки точности значений можно использовать следующие команды: *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 и *TruncTod* на стр. 2-757.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте для In[] массив, элементы которого являются структурами. В противном случае произойдет ошибка сборки.
- Типы данных *Key* и *Member* должны быть одинаковыми. При использовании разных типов данных произойдет ошибка сборки.
- Если в In[] будет передан элемент массива, будут обработаны все элементы, расположенные ниже переданного элемента.
- Если *Member* является вещественным числом, ожидаемый результат может быть не получен из-за ошибки округления, в зависимости от значения числа.
- Если *Key* является вещественным числом, не указывайте нечисловое значение для *Key*.
- Если *Size* = 0, то *Out* = ЛОЖЬ, а *Num* = 0. Значение InOutPos[] не изменится.
- Если *Mode* = _BIN_ASC или _BIN_DESC и при этом элементы массива In[] не отсортированы в порядке возрастания или убывания, правильный результат получен не будет. Прежде чем выполнять эту команду, отсортируйте элементы в порядке возрастания или убывания.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значения *Out*, InOutPos[] и *Num* не изменятся.
  a) Значение *Mode* находится за пределами допустимого диапазона.
  b) Значение *Size* приводит к выходу за область массива In[].
  c) *Member* не является членом структуры In[].
  d) Размер массива InOutPos[] меньше, чем число измерений в массиве In[].
  e) *Member* содержит строковое значение (STRING) и не завершается символом NULL.

RecRangeSearch

Команда RecRangeSearch производит поиск в массиве структур, используя указанный метод поиска. Ищутся элементы, содержащие значение, которое находится в заданном диапазоне.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
RecRangeSearch	Поиск диапазона записей	FUN		Out:=RecRangeSearch(In, Size, Member, MN, MX, Condition, Mode, InOutPos, Num);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для поиска	Вход	Массив структур для поиска	---	---	*1
Size	Количество элементов для поиска		Количество элементов массива для поиска	Зависит от типа данных.		1
Member	Член структуры для поиска		Член структуры In[] для поиска			*1
MN	Нижний предел условия поиска		Нижний предел условия поиска			
MX	Верхний предел условия поиска		Верхний предел условия поиска			
Condition	Условие поиска		Условие поиска	_EQ_BOTH, _EQ_MIN, _EQ_MAX, _NE_BOTH		_EQ_BOTH
Mode	Метод поиска		Метод поиска	_LINEAR, _BIN_ASC, _BIN_DESC		_LINEAR
InOutPos[] (массив)	Номера элементов, удовлетворяющих условиям поиска	Вход-выход	Номера элементов, удовлетворяющих условиям поиска	Зависит от типа данных.	---	---
Out	Результат поиска	Выход	ИСТИНА: есть элементы, соответствующие условиям поиска ЛОЖЬ: нет элементов, соответствующих условиям поиска	Зависит от типа данных.	---	---
Num	Число совпадений		Число совпадений			

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)	Следует указать массив структур.																			
Size							ОК													
Member						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	OK *1	OK *1	OK *1	OK *1	OK *1
	Следует указать такой же тип данных, как у члена структуры для поиска In[].																			
MN	Тип данных должен быть таким же, как у *Member*.																			
MX	Тип данных должен быть таким же, как у *Member*.																			
Condition	Сведения о перечислителях перечислимого типа _eSEARCH_CONDITION см. в разделе *Функция* на стр. 2-590.																			
Mode	Сведения о перечислителях перечислимого типа _eSEARCH_MODE см. в разделе *Функция* на стр. 2-590.																			
InOutPos[] (массив)							ОК													
Out	ОК																			
Num							ОК													

*1. При использовании модуля ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше можно указать следующий тип: TIME, DATE, TOD, DT и STRING.

Функция

Команда RecRangeSearch производит поиск среди *Size* элементов массива структур In[], т. е. среди элементов с In[0] по In[*Size*-1]. Ищутся элементы, в которых содержимое члена структуры *Member* (член структуры для поиска) удовлетворяет условию поиска.
Параметр *Condition* задает условие поиска, а параметр *Mode* задает метод поиска. Более подробно команда будет рассмотрена ниже.
Член структуры элемента массива In[], в котором должен производиться поиск, передается в качестве аргумента в параметр *Member*.
При обнаружении хотя бы одного элемента, отвечающего условиям поиска, значение переменной *Out* (результат поиска) меняется на ИСТИНА. Номер найденного элемента присваивается элементу массива InOutPos[0], а в переменную *Num* записывается количество найденных элементов. Если найдено несколько элементов с искомым значением, элементу InOutPos[0] присваивается наименьший номер найденного элемента массива In[].
Если ни одного элемента не будет найдено, выход *Out* будет находиться в состоянии ЛОЖЬ, а InOutPos[0] и *Num* будут равны 0.

Всегда присоединяйте номер элемента к входному параметру, который передается в In[], например: array[3].

Для параметра *Condition* используется перечислимый тип данных _eSEARCH_CONDITION. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_EQ_BOTH	*MN* ≤ *Member* ≤ *MX*
_EQ_MIN	*MN* ≤ *Member* < *MX*
_EQ_MAX	*MN* < *Member* ≤ *MX*

Перечислитель	Значение
_NE_BOTH	*MN* < *Member* < *MX*

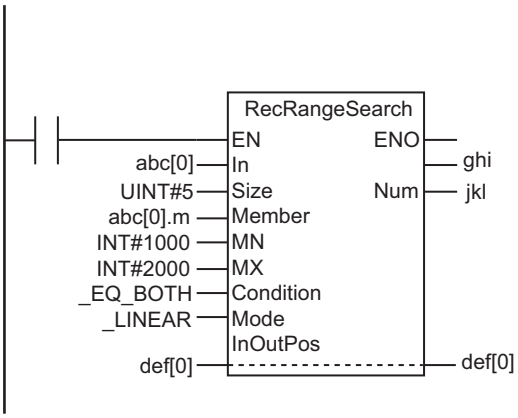
Для параметра *Mode* используется перечислимый тип данных eSEARCH_MODE. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_LINEAR	Линейный поиск
_BIN_ASC	Бинарный поиск в порядке возрастания
_BIN_DESC	Бинарный поиск в порядке убывания

При линейном поиске поиск выполняется по порядку, начиная с первого элемента массива In[].

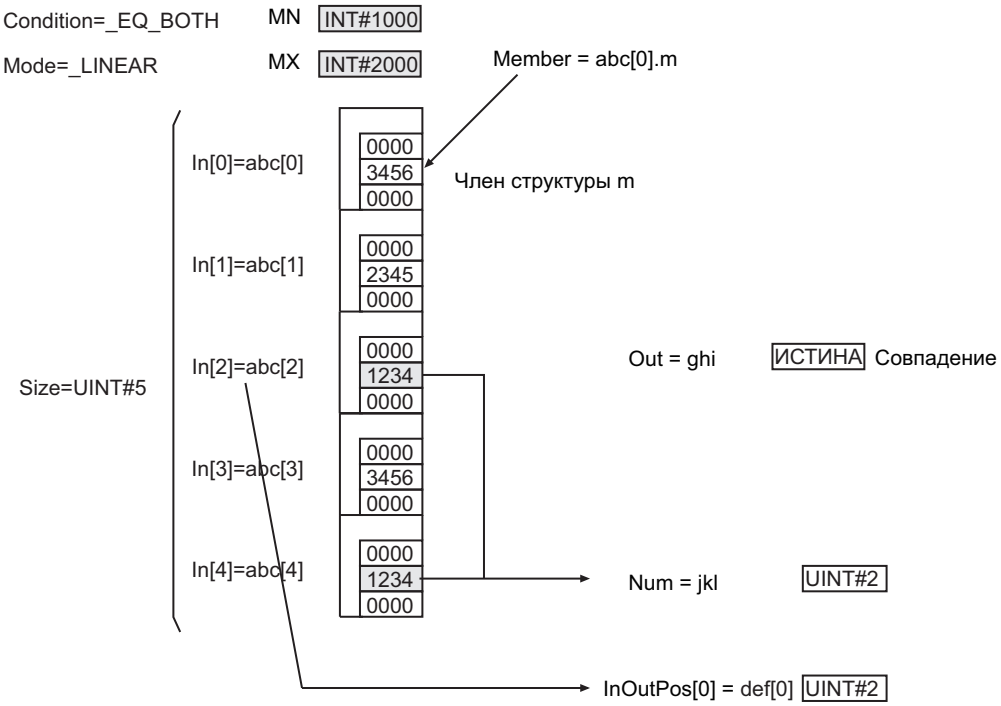
Ниже показан пример для случая, когда *Size* = UINT#5, *MN* = INT#1000, *MX* = INT#2000, *Condition* = _EQ_BOTH, а *Mode* = _LINEAR.

LD



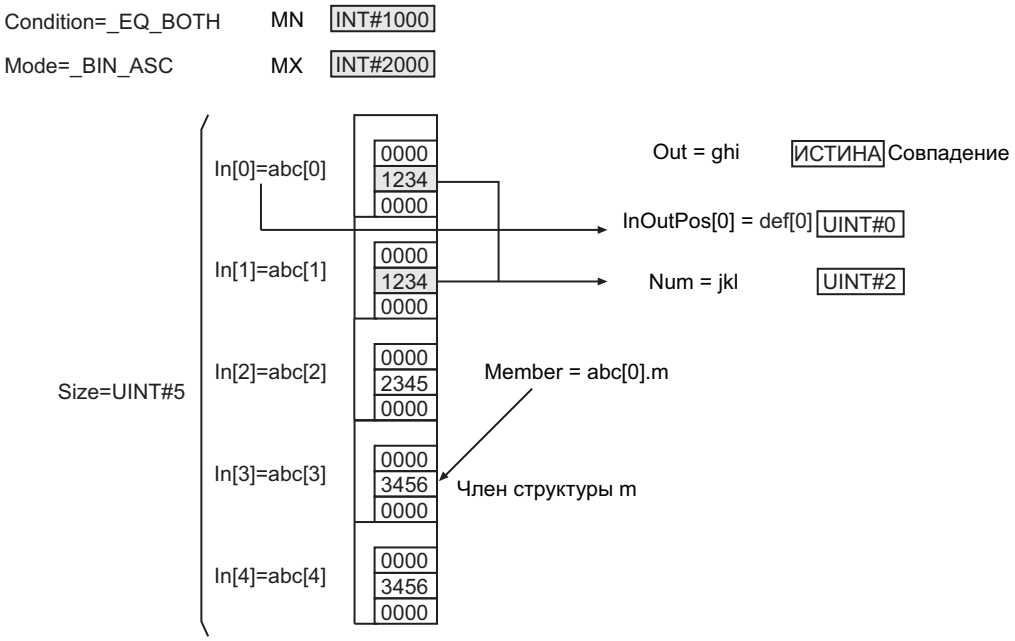
ST

```
ghi:=RecRangeSearch(abc[0], UINT#5, abc[0].m, INT#1000,
                    INT#2000, _EQ_BOTH, _LINEAR,
                    def[0], jkl);
```



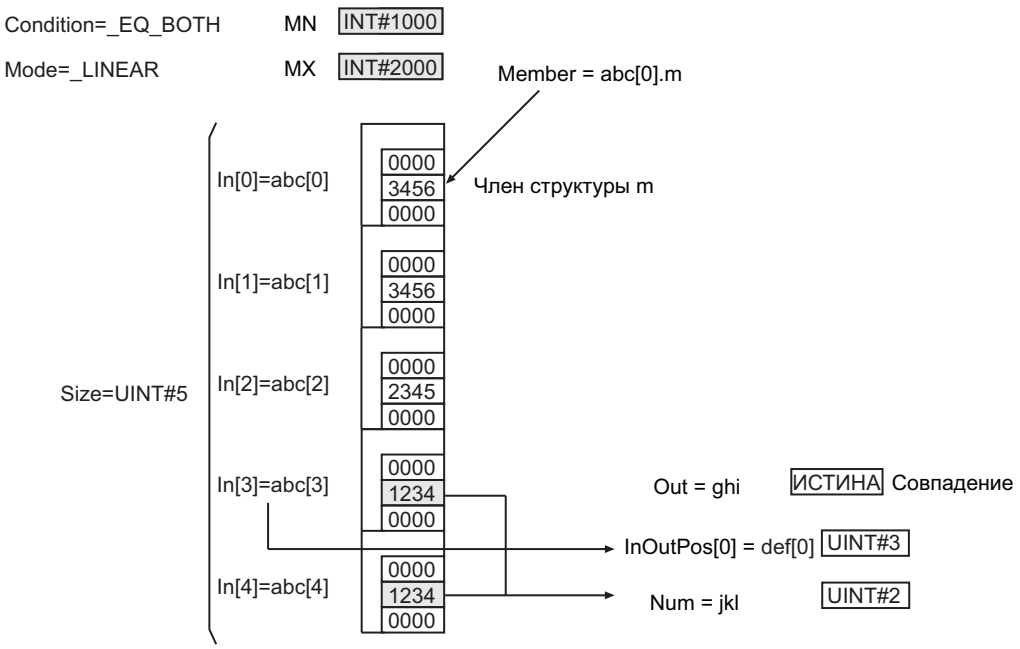
При бинарном поиске в порядке возрастания элементы массива, передаваемого в качестве входного параметра в переменную In[], должны быть заранее отсортированы в порядке возрастания. Затем выполняется команда для выполнения бинарного поиска.

В следующем примере используются те же исходные данные, что в примере выше, однако элементы массива отсортированы в порядке возрастания. Производится бинарный поиск, результаты которого показаны ниже.



При бинарном поиске в порядке убывания элементы массива, передаваемого в качестве входного параметра в переменную In[], должны быть заранее отсортированы в порядке убывания. Затем выполняется команда для выполнения бинарного поиска.

В следующем примере используются те же исходные данные, что и в предыдущем примере, однако элементы массива отсортированы в порядке убывания. Производится бинарный поиск, результаты которого показаны ниже.



Дополнительная информация

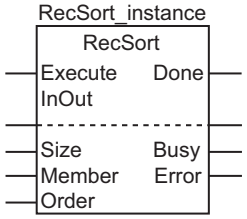
- Массив In[] может быть членом структуры более высокого уровня.
  Пример: In[0]=str0.str1[0]
- In[] может быть массивом с двумя или более измерениями. Если In[] является двумерным массивом, номер соответствующего условиям поиска элемента, найденного в первом измерении, присваивается элементу InOutPos[0], а номер элемента, найденного во втором измерении, присваивается элементу InOutPos[1].
- Если In[] является трехмерным массивом, номер соответствующего условиям поиска элемента, найденного в первом измерении, присваивается элементу InOutPos[0]; номер элемента, найденного во втором измерении, присваивается элементу InOutPos[1]; номер элемента, найденного в третьем измерении, присваивается элементу InOutPos[2].
- При поиске значения типа TIME, DT или TOD необходимо установить одинаковую точность для значений параметров *Member*, *MN* и *MX*. Для установки точности значений можно использовать следующие команды: *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 и *TruncTod* на стр. 2-757.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте для переменных *Member*, *MN* и *MX* такой же тип данных, как у искомого члена структуры In[]. В противном случае произойдет ошибка сборки.
- Используйте для In[] массив, элементы которого являются структурами. В противном случае произойдет ошибка сборки.
- Если в In[] будет передан элемент массива, будут обработаны все элементы, расположенные ниже переданного элемента.
- Если *Member* является вещественным числом, результат может быть неверным из-за ошибки округления, в зависимости от значения числа.
- Если параметр *MN* или *MX* является вещественным числом, не указывайте для него нечисловое значение.
- Если *Size* = 0, то *Out* = ЛОЖЬ, а *Num* = 0. Значение InOutPos[] не изменяется.
- Если *Mode* = _BIN_ASC или _BIN_DESC и при этом элементы массива In[] не отсортированы в порядке возрастания или убывания, правильный результат получен не будет. Прежде чем выполнять эту команду, отсортируйте элементы в порядке возрастания или убывания.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значения *Out*, InOutPos[] и *Num* не изменятся.
  a) Значение *MN* больше значения *MX*.
  b) Значение *Condition* находится за пределами допустимого диапазона.
  c) Значение *Mode* находится за пределами допустимого диапазона.
  d) Значение *Size* приводит к выходу за область массива In[].
  e) *Member* не является членом структуры In[].
  f) Размер массива InOutPos[] меньше, чем число измерений в массиве In[].

RecSort

Команда RecSort сортирует элементы массива структур.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
RecSort	Сортировка записей	FB		RecSort_instance(Execute, InOut, Size, Member, Order, Done, Busy, Error);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Size	Количество элементов для сортировки	Вход	Количество элементов массива для сортировки	Зависит от типа данных.	---	1
Member	Член структуры для сортировки		Член структуры In[] для сортировки			*1
Order	Порядок сортировки		Порядок сортировки	_ASC, _DESC		_ASC
InOut[] (массив)	Массив для сортировки	Вход-выход	Массив структур для сортировки	---	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Size							OK													
Member						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK *1	OK *1	OK *1	OK *1	OK *1
	Следует указать такой же тип данных, как у члена структуры для сортировки InOut[].																			
Order	Сведения о перечислителях перечислимого типа _eSORT_ORDER см. в разделе *Функция* на стр. 2-595.																			
InOut[] (массив)	Следует указать массив структур.																			

*1. При использовании модуля ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше можно указать следующий тип: TIME, DATE, TOD, DT и STRING.

Функция

Когда значение *Execute* = ИСТИНА, команда RecSort сортирует *Size* элементов массива InOut[] (массив структур) (элементы с InOut[0] по InOut[*Size*-1]) по значениям *Member* (член структуры для сортировки). Параметр *Order* задает порядок сортировки.
Член структуры элемента массива In[], по которому должна производиться сортировка, передается в качестве аргумента в параметр *Member*.

Всегда присоединяйте номер элемента к входному-выходному параметру, который передается в переменную InOut[], например: array[3].

Для параметра *Order* используется перечислимый тип данных _eSORT_ORDER. Значения перечислителей приведены в таблице ниже.

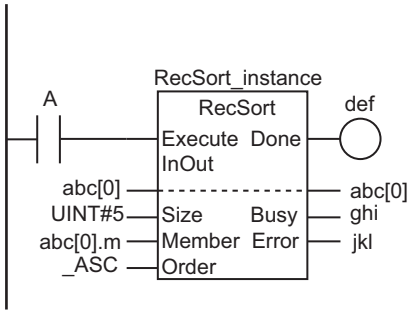
Перечислитель	Значение
_ASC	В порядке возрастания
_DESC	В порядке убывания

Как соотносятся между собой значения, не являющиеся целыми или вещественными числами, поясняется в таблице ниже.

Тип данных	Соотношение
TIME	За большее принимается значение, которое больше в числовом выражении.
DATE, TOD или DT	За большее принимается более поздняя дата или время суток.
STRING	Применяются те же правила, что и для команд *LTascii, LEascii, GTascii u GEascii* на стр. 2-122. Подробные сведения см. на указанных страницах.

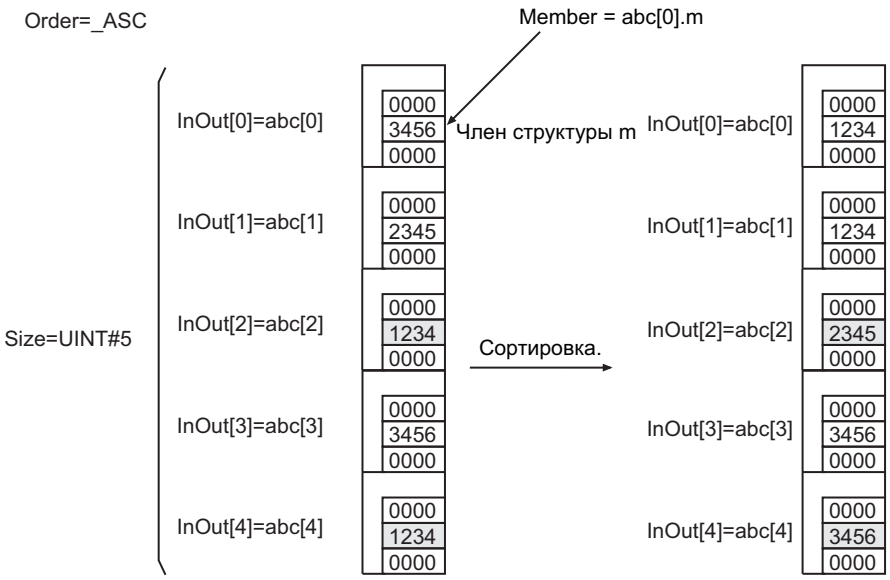
Ниже показан пример для случая, когда *Size* = UINT#5, а *Order* = _ASC.

LD



ST

```
RecSort_instance(A, abc[0], UINT#5, abc[0].m, _ASC, def, ghi, jkl);
```



Дополнительная информация

- Если во время выполнения этой команды будет прервано питание, содержимое InOut может быть повреждено. Чтобы поврежденные данные можно было восстановить, перед выполнением этой команды каждый раз следует создавать резервную копию массива InOut[].
  См. раздел *Пример программы* на стр. 2-597.
- При сортировке значений типа TIME, DT или TOD необходимо установить одинаковую точность для всех значений *Member*. Для установки точности значений можно использовать команды *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 или *TruncTod* на стр. 2-757.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте для InOut[] массив, элементы которого являются структурами. В противном случае произойдет ошибка сборки.
- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если *Member* является вещественным числом, результат может быть неверным из-за ошибки округления, в зависимости от значения числа.
- Если в InOut[] будет передан элемент массива, будут обработаны все последующие элементы.
- Если *Size* = 0, выход *Done* будет содержать ИСТИНА, а значение InOut[] не изменится.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение *Order* находится за пределами допустимого диапазона.
  b) Значение *Size* приводит к выходу за область массива InOut[].
  c) *Member* не является членом структуры InOut[].
  d) *Member* содержит строковое значение (STRING) и не завершается символом NULL.

Пример программы

В этом примере команда RecSort сортирует массив Abc[] структур *MyStr* в порядке возрастания. Сортировка выполняется на основе значения члена структуры Abc[].m.
Чтобы предотвратить потерю данных из-за прерывания питания во время обработки, перед сортировкой создается резервная копия массива Abc[] в переменной с именем Abc_backup[]. Если произойдет сбой питания, содержимое Abc[] будет восстановлено из резервной копии Abc_backup[] и операция сортировки будет выполнена повторно.

Определения глобальных переменных

● Типы данных

Переменная	Тип данных	Комментарий
MyStr	STRUCT	Структура
l	BOOL	Член структуры
m	INT	Член структуры
n	REAL	Член структуры

● Глобальные переменные

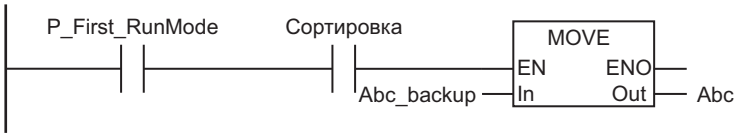
Переменная	Тип данных	Начальное значение	Сохранение	Комментарий
Abc	ARRAY[0..4] OF MyStr	[5((l:=FALSE,m:=0,n:=0.0))]	☑	Массив для сортировки
Abc_backup	ARRAY[0..4] OF MyStr	[5((l:=FALSE,m:=0,n:=0.0))]	☑	Резервная копия Abc[]

Программа на языке LD

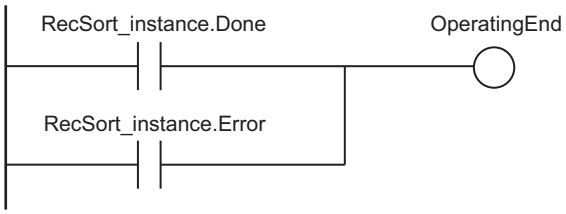
Внутренние переменные	Переменная	Тип данных	Начальное значение	Сохранение	Комментарий
	Sorting	BOOL	ЛОЖЬ	☑	Обработка (сохраняется)
	OperatingEnd	BOOL	ЛОЖЬ	☐	Обработка завершена
	Trigger	BOOL	ЛОЖЬ	☐	Условие выполнения
	Operating	BOOL	ЛОЖЬ	☐	Обработка
	RS_instance	RS		☐	
	RecSort_instance	RecSort		☐	

Внешние переменные	Переменная	Тип данных	Комментарий
	Abc	ARRAY[0..4] OF MyStr	Массив для сортировки
	Abc_backup	ARRAY[0..4] OF MyStr	Резервная копия Abc[]

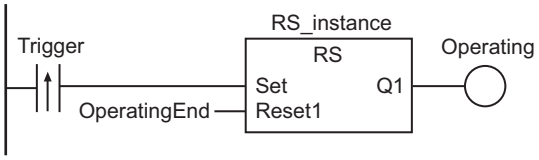
Восстановление ***Abc[]*** из резервной копии ***Abc_backup[]*** после прерывания питания.



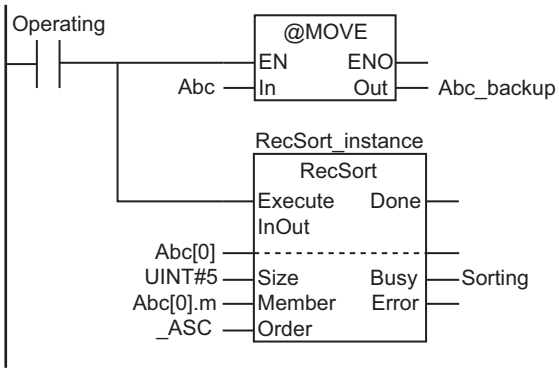
Проверка завершения выполнения команды RecSort.



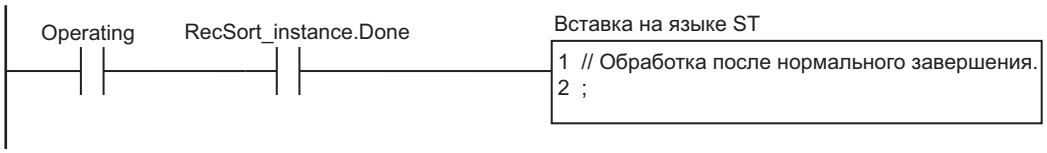
Прием условия выполнения.



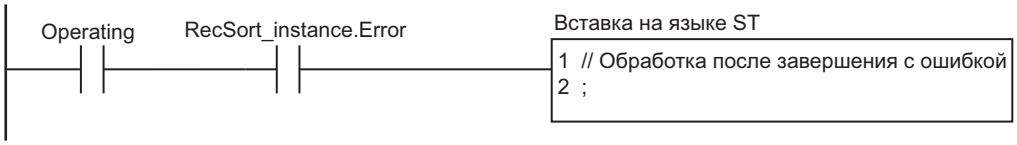
Создание резервной копии и выполнение команды RecSort.



Обработка после нормального завершения.



Обработка после завершения с ошибкой



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Сохранение	Комментарий
	Sorting	BOOL	ЛОЖЬ	☑	Обработка (сохраняется)
	Trigger	BOOL	ЛОЖЬ	☐	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	☐	Значение *Trigger* в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	☐	Обработка началась
	Operating	BOOL	ЛОЖЬ	☐	Обработка
	RS_instance	RS		☐	
	RecSort_instance	RecSort		☐	

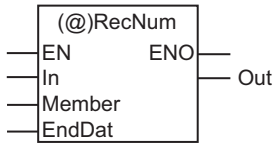
Внешние переменные	Переменная	Тип данных	Комментарий
	Abc	ARRAY[0..4] OF MyStr	Массив для сортировки
	Abc_backup	ARRAY[0..4] OF MyStr	Резервная копия Abc[]

```
// Восстановление Abc[] из резервной копии Abc_backup[] после прерывания питания.
IF ( (P_First_RunMode = TRUE) AND (Sorting = TRUE) ) THEN
  Abc:=Abc_backup;
END_IF;

// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) ) THEN
  OperatingStart:=TRUE;
  Operating :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация команды RecSort.
IF (OperatingStart=TRUE) THEN
  Abc_backup:=Abc;
  RecSort_instance(
    Execute:=FALSE, // Start condition
    InOut :=Abc[0], // Sort array
    Member :=Abc[0].m); // Member to sort
  OperatingStart:=FALSE;
END_IF;

// Выполнение команды RecSort.
IF (Operating=TRUE) THEN
  RecSort_instance(
```

```
    Execute:=TRUE,
    InOut :=Abc[0],
    Size :=UINT#5,
    Member :=Abc[0].m,
    Order :=_ASC,
    Busy =>Sorting);

  IF (RecSort_instance.Done=TRUE) THEN
    // Обработка после нормального завершения.
    Operating:=FALSE;
  END_IF;

  IF (RecSort_instance.Error=TRUE) THEN
    // Обработка после завершения с ошибкой.
    Operating:=FALSE;
  END_IF;
END_IF;
```

RecNum

Команда RecNum определяет количество записей в массиве структур, предшествующих искомым («конечным») данным.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
RecNum	Получить количество записей	FUN		Out:=RecNum(In, Member, EndDat);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для обработки	Вход	Массив структур для обработки	---	---	*1
Member	Член структуры для обработки		Член структуры In[] для обработки	Зависит от типа данных.		
EndDat	Конечные данные		Конечные данные			
Out	Количество записей	Выход	Количество записей	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)	Следует указать массив структур.																			
Member	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК *1	ОК *1	ОК *1	ОК *1	ОК
	Также допускается указывать перечисления.*2																			
	Тип данных должен быть таким же, как у членов структуры для обработки в In[].																			
EndDat	Тип данных должен быть таким же, как у *Member*.																			
Out							ОК													

*1. При использовании модуля ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше можно указать значение следующего типа: TIME, DATE, TOD и DT.

*2. Чтобы можно было указывать перечисления, требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

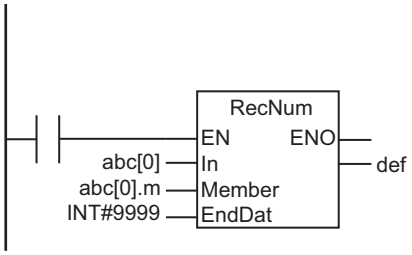
Функция

Команда RecNum принимает массив структур In[] и производит в нем поиск элемента, содержащего член структуры *Member* (член структуры для обработки) со значением *EndDat* (конечные данные). После этого в переменную *Out* записывается количество элементов (записей), расположенных перед найденным элементом (т. е. элементом, значение указанного члена структуры которого совпадает с *EndDat* (конечные данные)).
Член структуры элемента массива In[], по которому должен производиться поиск, передается в качестве аргумента в параметр *Member*.

Всегда присоединяйте номер элемента к входному параметру, который передается в In[], например: array[3].

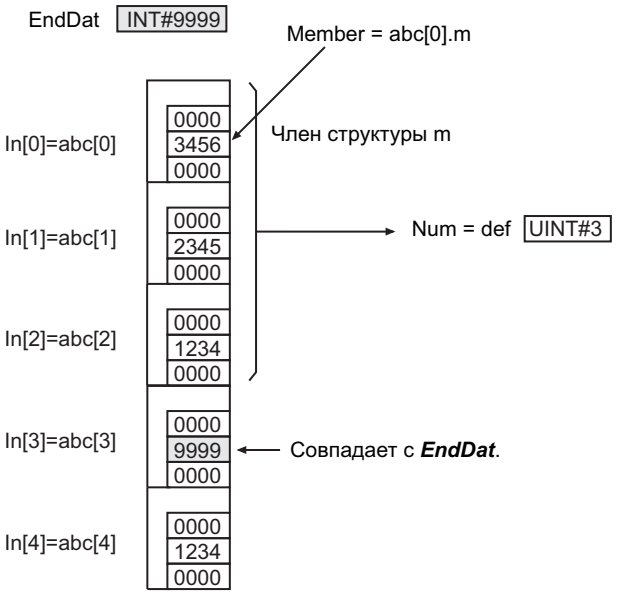
Ниже показан пример программы, в котором *EndDat* = INT#9999.

LD



ST

```
def:=RecNum(abc[0], abc[0].m, INT#9999);
```



Дополнительная информация

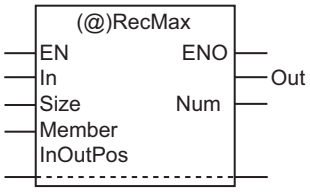
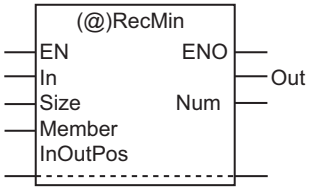
- Массив In[] может быть членом структуры более высокого уровня.
  Пример: In[0]=str0.str1[0]
- При поиске значения типа TIME, DT или TOD необходимо установить одинаковую точность для значений параметров *Member* и *EndDat*. Для установки точности значений можно использовать следующие команды: *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 и *TruncTod* на стр. 2-757.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте для In[] массив, элементы которого являются структурами. В противном случае произойдет ошибка сборки.
- Типы данных *Member* и *EndDat* должны быть одинаковыми. При использовании разных типов данных произойдет ошибка сборки.
- Если ни один из членов структуры в In[] не содержит значения *EndDat*, в *Out* записывается общее количество элементов в массиве In[].
- Если *Member* является вещественным числом, результат может быть неверным из-за ошибки округления, в зависимости от значения числа.
- Если параметр *EndDat* является вещественным числом, не указывайте для *EndDat* нечисловое значение.
- Если в In[] будет передан элемент массива, будут обработаны все элементы, расположенные ниже переданного элемента.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) *Member* не является членом структуры In[].
  b) *Member* содержит строковое значение (STRING) и не завершается символом NULL.

RecMax и RecMin

RecMax : Ищет в массиве структур максимальное значение указанного члена структуры.
RecMin : Ищет в массиве структур минимальное значение указанного члена структуры.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
RecMax	Поиск записи с максимальным значением	FUN		Out:=RecMax(In, Size,Member, InOutPos, Num);
RecMin	Поиск записи с минимальным значением	FUN		Out:=RecMin(In, Size, Member, InOutPos, Num);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для поиска	Вход	Массив структур для поиска	---	---	*1
Size	Количество элементов для поиска		Количество элементов массива для поиска	Зависит от типа данных.	---	1
Member	Член структуры для поиска		Член структуры In[] для поиска			*1
InOutPos[] (массив)	Номер найденного элемента	Вход-выход	Номер найденного элемента	Зависит от типа данных.	---	---
Out	Результат поиска	Выход	Результат поиска	Зависит от типа данных.	---	---
Num	Количество найденных элементов		Количество найденных элементов			

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)	Следует указать массив структур.																			
Size							OK													

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Member						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK *1	OK *1	OK *1	OK *1	OK *1
	Следует указать такой же тип данных, как у члена структуры для поиска In[].																			
InOutPos[] (массив)							OK													
Out						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK *1	OK *1	OK *1	OK *1	OK *1
Num							OK													

*1. При использовании модуля ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше можно указать следующий тип: TIME, DATE, TOD, DT и STRING.

Функция

Эти команды ищут минимальное или максимальное значение *Member* (член структуры для поиска) среди *Size* элементов (т. е. с In[0] по In[*Size*-1]) массива структур In[].
Один из членов структуры элемента массива In[] передается в качестве аргумента в параметр *Member*.
Номер найденного элемента с минимальным или максимальным значением присваивается элементу массива InOutPos[0], а в переменную *Num* записывается количество найденных элементов с таким значением. Если найдено несколько элементов массива In[] с таким значением, элементу массива InOutPos[0] присваивается наименьший номер найденного элемента.

Всегда присоединяйте номер элемента к входному параметру, который передается в In[], например: array[3].

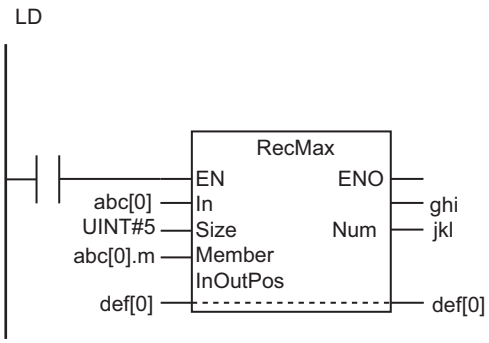
Как соотносятся между собой значения, не являющиеся целыми или вещественными числами, поясняется в таблице ниже.

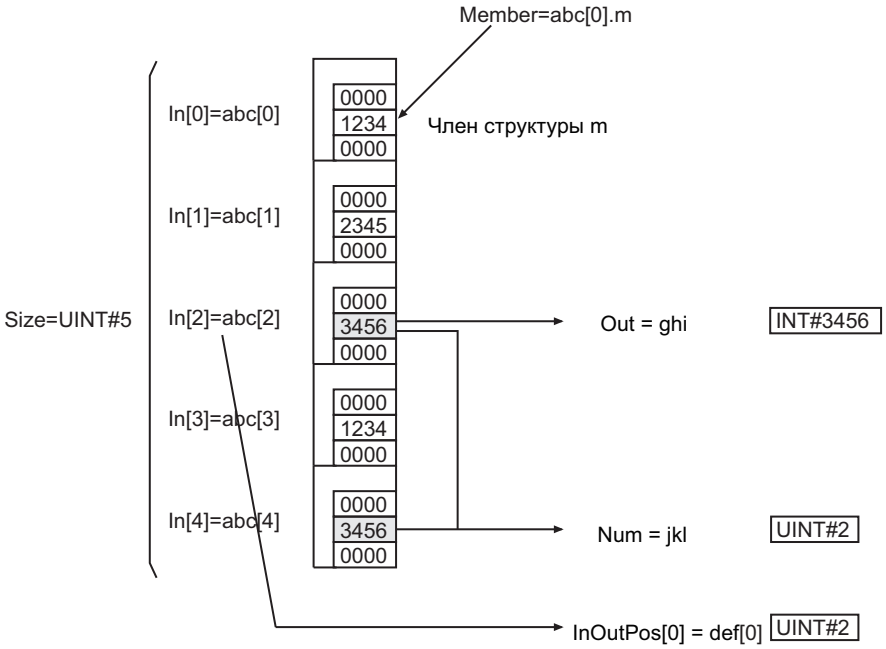
Тип данных	Соотношение
TIME	За большее принимается значение, которое больше в числовом выражении.
DATE, TOD или DT	За большее принимается более поздняя дата или время суток.
STRING	Действуют те же правила, что и для команд *LTascii, LEascii, GTascii и GEascii* на стр. 2-122. Подробную информацию см. на указанной странице.

RecMax

Команда RecMax ищет максимальное значение члена структуры для поиска и записывает найденное максимальное значение в переменную *Out* (результат поиска).

Ниже показан пример для команды RecMax, в котором *Size* = UINT#5.





RecMin

Команда RecMin ищет минимальное значение члена структуры для поиска и записывает найденное минимальное значение в переменную *Out* (результат поиска).

Дополнительная информация

- Массив In[] может быть членом структуры более высокого уровня.
  Пример: In[0]=str0.str1[0]
- In[] может быть массивом с двумя или более измерениями. Если In[] является двумерным массивом, номер соответствующего условиям поиска элемента, найденного в первом измерении, присваивается элементу InOutPos[0], а номер элемента, найденного во втором измерении, присваивается элементу InOutPos[1].
- Если In[] является трехмерным массивом, номер соответствующего условиям поиска элемента, найденного в первом измерении, присваивается элементу InOutPos[0]; номер элемента, найденного во втором измерении, присваивается элементу InOutPos[1]; номер элемента, найденного в третьем измерении, присваивается элементу InOutPos[2].
- При поиске значения типа TIME, DT или TOD необходимо установить одинаковую точность для всех значений *Member*. Для установки точности значений можно использовать следующие команды: *TruncTime* на стр. 2-748, *TruncDt* на стр. 2-753 и *TruncTod* на стр. 2-757.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если для переменных *Member* и *Out* используются разные типы данных, они должны входить в указанную ниже группу данных. Кроме того, диапазон допустимых значений *Out* должен вмещать в себя диапазон допустимых значений *Member*.
  a) USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL и LREAL
- Если *Member* является вещественным числом, результат может быть неверным из-за ошибки округления, в зависимости от значения числа.
- Если в In[] будет передан элемент массива, будут обработаны все элементы, расположенные ниже переданного элемента.
- Если *In* является перечислением, всегда используйте переменную в качестве входного параметра, передаваемого в *In*. При передаче константы произойдет ошибка сборки.
- Если значение *Size* равно 0, то значения *Out* и *Num* также равны 0. Если *Member* содержит строковое значение (STRING), а *Size* = 0, то *Out* представляет собой текстовую строку, содержащую только нулевые символы (NULL). Значения InOutPos[] при этом не изменяется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а значения *Out*, InOutPos[] и *Num* не изменятся.
  a) Значение *Size* приводит к выходу за область массива In[].
  b) *Member* не является членом структуры In[].
  c) Размер массива InOutPos[] меньше, чем число измерений в массиве In[].
  d) *Member* содержит строковое значение (STRING) и не завершается символом NULL.

Команды вычисления контрольной суммы

Команда	Имя	Стр.
StringSum	Расчет контрольной суммы	стр. 2-610
StringLRC	Расчет LRC текстовой строки	стр. 2-612
StringCRCCCITT	Расчет CRC-CCITT текстовой строки	стр. 2-614
StringCRC16	Расчет CRC-16 текстовой строки	стр. 2-616
AryLRC_**	Группа расчета LRC массива	стр. 2-618
AryCRCCCITT	Расчет CRC-CCITT массива	стр. 2-620
AryCRC16	Расчет CRC-16 массива	стр. 2-622

StringSum

Команда StringSum вычисляет контрольную сумму для текстовой строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
StringSum	Расчет контрольной суммы	FUN	(@)StringSum EN ENO In Out Size	Out:=StringSum(In, Size);

Переменные

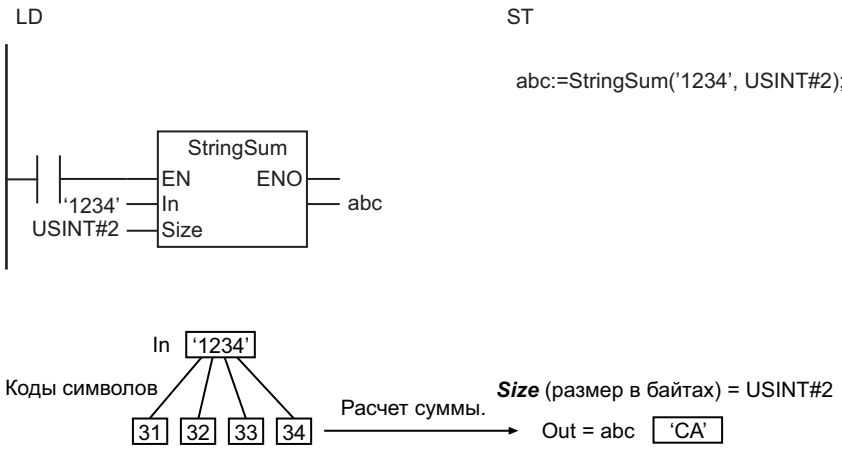
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Текстовая строка для обработки	Вход	Текстовая строка для обработки	Зависит от типа данных.	---	''
Size	Размер в байтах		Размер контрольной суммы в байтах	1 или 2	Байты	1
Out	Checksum	Выход	Checksum	Количество байтов, указанное параметром *Size*	Байты	---

	Логический тип	Битовые строки				Целое число								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Size						OK														
Out																				OK

Функция

Команда StringSum вычисляет контрольную сумму *In* (текстовая строка для обработки). Контрольная сумма в *Out* будет содержать количество байтов, указанное параметром *Size* (размер в байтах).
Out представляет из себя текстовую строку с шестнадцатеричными цифрами и пустым символом (NULL) в конце.

В приведенном ниже примере *In* = «1234», а *Size* = USINT#2.



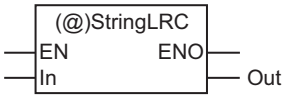
Если бы *Size* в примере выше имело значение USINT#1, то *Out* было бы равно «A».

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если количество цифр в сумме кодов символов в *In* превышает количество цифр, определенное параметром *Size*, старшие цифры отбрасываются.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Size* находится за пределами допустимого диапазона.
  b) Количество байтов в *In* равно 0 (т.е. строка содержит только пустые символы (NULL)).

StringLRC

Команда StringLRC вычисляет значение LRC (поперечный контроль четности).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
StringLRC	Расчет LRC текстовой строки	FUN		Out:=StringLRC(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Текстовая строка для обработки	Вход	Текстовая строка для обработки	Зависит от типа данных.	---	"
Out	Значение LRC	Выход	Значение LRC	Макс. 3 байта (два однобайтовых буквенно-цифровых символа + последний символ NULL)	---	---

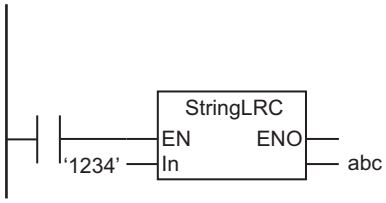
	Логический тип	Битовые строки				Целое число								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Out																				OK

Функция

Команда StringLRC вычисляет значение LRC (поперечный контроль четности) текстовой строки *In* (текстовая строка для обработки). Значение LRC представляет собой исключающее логическое ИЛИ кодов символов текстовой строки в *In*.
Значение LRC в *Out* представляет из себя текстовую строку с шестнадцатеричными цифрами и пустым символом (NULL) в конце.

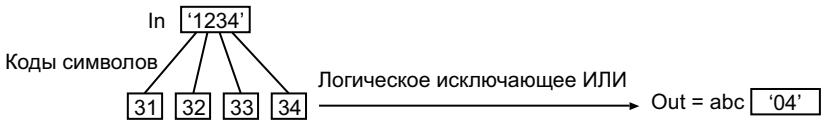
В приведенном ниже примере *In* = «1234».

LD



ST

```
abc:=StringLRC('1234');
```



Меры предосторожности для обеспечения надлежащей эксплуатации

В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Количество байтов в *In* равно 0 (т.е. строка содержит только пустые символы (NULL)).

StringCRCCCITT

Команда StringCRCCCITT вычисляет значение CRC-CCITT, используя метод XMODEM.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
StringCRCCCITT	Расчет CRC-CCITT текстовой строки	FUN	(@)StringCRCCCITT EN — ENO In Initial OutOrder — Out	Out:=StringCRCCCITT(In, Initial, OutOrder);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Текстовая строка для обработки	Вход	Текстовая строка для обработки	Зависит от типа данных.	---	"
Initial	Начальное значение		Начальное значение CRC-CCITT			0
OutOrder	Порядок байтов		Порядок обработки байтов в *In*	_LOW_HIGH, _HIGH_LOW		_HIGH_LOW
Out	Значение CRC-CCITT	Выход	Значение CRC-CCITT	5 байтов (4 однобайтовых буквенно-цифровых символа + последний символ NULL)	---	---

	Логический тип	Битовые строки				Целое число								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				ОК
Initial			ОК																	
OutOrder	Сведения о перечислителях перечислимого типа _eBYTE_ORDER см. в разделе *Функция* на стр. 2-614.																			
Out																				ОК

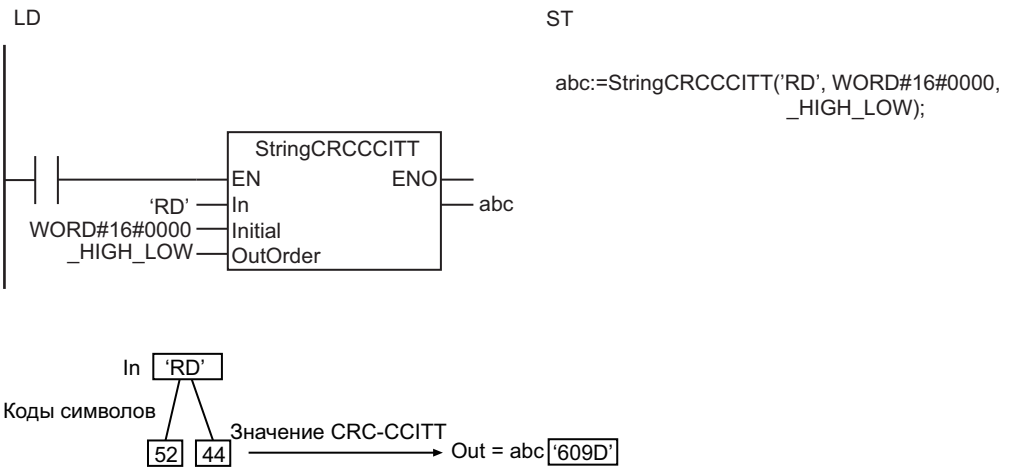
Функция

Команда StringCRCCCITT вычисляет значение CRC-CCITT текстовой строки *In* (текстовая строка для обработки), используя метод XMODEM.
Значение CRC-CCITT в *Out* представляет из себя текстовую строку с шестнадцатеричными цифрами и пустым символом (NULL) в конце.

В параметре *Initial* следует указать начальное значение для расчета значения CRC-CCITT. В параметре *OutOrder* задаётся порядок обработки байтов.
Для параметра *OutOrder* используется перечислимый тип данных _eBYTE_ORDER. Значения перечислителей приведены в таблице ниже.

Перечислители	Значение
_LOW_HIGH	Младший байт первым, старший байт последним
_HIGH_LOW	Старший байт первым, младший байт последним

В приведенном ниже примере *In* = «RD», *Initial* = WORD#16#0000, а *OutOrder* = _HIGH_LOW.



Меры предосторожности для обеспечения надлежащей эксплуатации

В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Значение *OutOrder* находится за пределами допустимого диапазона.
- Количество байтов в *In* равно 0 (т.е. строка содержит только пустые символы (NULL)).

StringCRC16

Команда StringCRC16 вычисляет значение CRC-16, используя метод MODBUS.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
StringCRC16	Расчет CRC-16 текстовой строки	FUN	(@)StringCRC16 EN ENO In Out Initial OutOrder	Out:=StringCRC16(In, Initial, OutOrder);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Текстовая строка для обработки	Вход	Текстовая строка для обработки	Зависит от типа данных.	---	"
Initial	Начальное значение		Начальное значение CRC-16			16#FFFF
OutOrder	Порядок байтов		Порядок обработки байтов в *In*	_LOW_HIGH, _HIGH_LOW		_LOW_HIGH
Out	Значение CRC-16	Выход	Значение CRC-16	5 байтов (4 однобайтовых буквенно-цифровых символа + последний символ NULL)	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Initial			OK																	
OutOrder	Сведения о перечислителях перечислимого типа _eBYTE_ORDER см. в разделе *Функция* на стр. 2-616.																			
Out																				OK

Функция

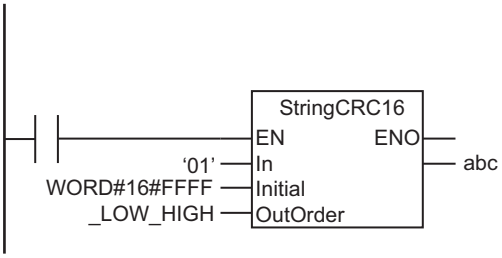
Команда StringCRC16 вычисляет значение CRC-16 текстовой строки *In* (текстовая строка для обработки), используя метод MODBUS.
Значение CRC-16 в *Out* представляет из себя текстовую строку с шестнадцатеричными цифрами и пустым символом (NULL) в конце.

В параметре *Initial* следует указать начальное значение для расчета значения CRC-16. В параметре *OutOrder* задаётся порядок обработки байтов.
Для параметра *OutOrder* используется перечислимый тип данных _eBYTE_ORDER. Значения перечислителей приведены в таблице ниже.

Перечислители	Значение
_LOW_HIGH	Младший байт первым, старший байт последним
_HIGH_LOW	Старший байт первым, младший байт последним

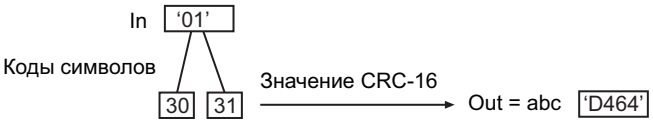
В приведенном ниже примере *In* = «01», *Initial* = WORD#16#FFFF, а *OutOrder* = _LOW_HIGH.

LD



ST

```
abc:=StringCRC16('01', WORD#16#FFFF,
                 _LOW_HIGH);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Значение *OutOrder* находится за пределами допустимого диапазона.
- Количество байтов в *In* равно 0 (т.е. строка содержит только пустые символы (NULL)).

AryLRC_**

Команды AryLRC_** вычисляют значение LRC массива.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryLRC_**	Группа расчета LRC массива	FUN	(@)AryLRC_** EN ENO In Out Size "**" — тип данных, относящийся к битовым строкам.	Out:=AryLRC_**(In, Size); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для обработки	Вход	Массив для обработки	Зависит от типа данных.	---	*1
Size	Количество элементов для обработки		Количество элементов массива In[]			1
Out	Значение LRC	Выход	Значение LRC	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)		ОК	ОК	ОК	ОК															
Size							ОК													
Out	Тип данных должен быть таким же, как у In[].																			

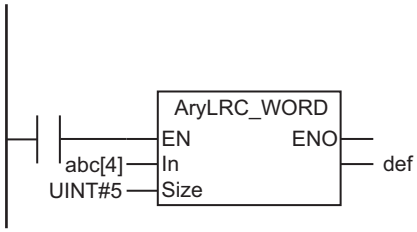
Функция

Команды AryLRC_** вычисляют значение LRC (исключающее логическое *ИЛИ*) для *Size* элементов массива In[] (массив для обработки), начиная с In[0].
Имя команды определяется типом данных In[]. Например, если In[] относится к типу данных WORD, команда будет иметь имя AryLRC_WORD.

Всегда присоединяйте номер элемента к входному-выходному параметру, который передается в In[], например: array[3].

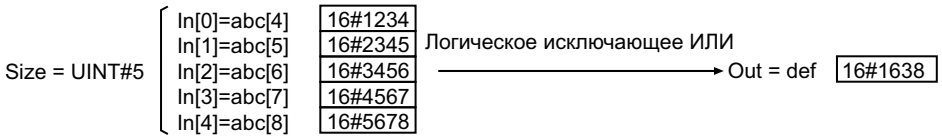
Ниже показан пример использования команды AryLRC_WORD для случая, когда *Size* = UINT#5.

LD



ST

```
def:=AryLRC_WORD(abc[4], UINT#5);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

- Используйте одинаковый тип данных для In[] и *Out*.
- Если значение *Size* равно 0, то значение *Out* также равно 16#00.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Size* приводит к выходу за область массива In[].

AryCRCCCITT

Команда AryCRCCCITT вычисляет значение CRC-CCITT, используя метод XMODEM.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryCRCCCITT	Расчет CRC-CCITT массива	FUN	(@)AryCRCCCITT EN In Size Initial OutOrder ENO Out	Out:=AryCRCCCITT(In, Size, Initial, OutOrder);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для обработки	Вход	Массив для обработки	Зависит от типа данных.	---	*1
Size	Количество элементов для обработки		Количество элементов массива In[]			1
Initial	Начальное значение		Начальное значение CRC-CCITT			0
OutOrder	Порядок байтов		Порядок обработки байтов в *In*	_LOW_HIGH, _HIGH_LOW		_HIGH_LOW
Out	Значение CRC-CCITT	Выход	Значение CRC-CCITT	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)		ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	
Size							ОК													
Initial			ОК																	
OutOrder	Сведения о перечислителях перечислимого типа _eBYTE_ORDER см. в разделе *Функция* на стр. 2-620.																			
Out			ОК																	

Функция

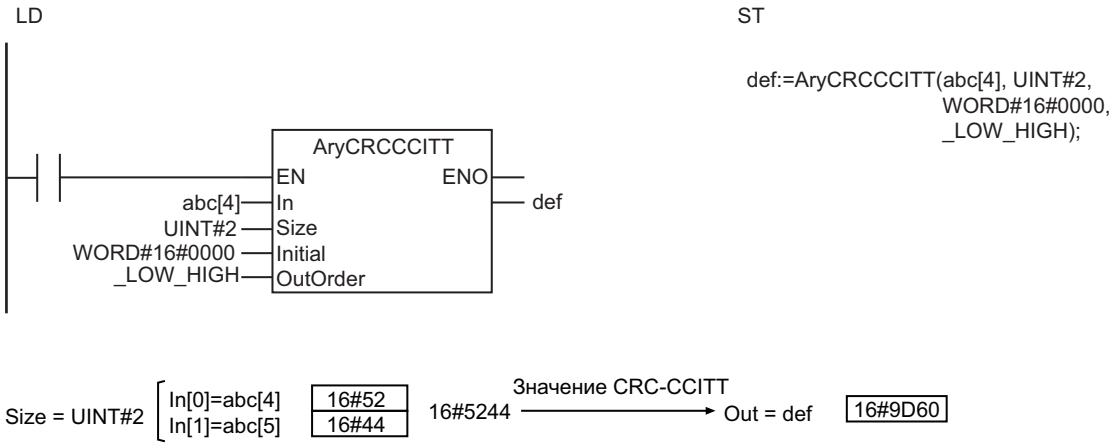
Команда AryCRCCCITT вычисляет значение CRC-CCITT для *Size* элементов массива In[] (массив для обработки), начиная с элемента In[0]. Используется метод XMODEM.

В параметре *Initial* следует указать начальное значение для расчета значения CRC-CCITT. В параметре *OutOrder* задаётся порядок обработки байтов.
Для параметра *OutOrder* используется перечислимый тип данных _eBYTE_ORDER. Значения перечислителей приведены в таблице ниже.

Перечислители	Значение
_LOW_HIGH	Младший байт первым, старший байт последним
_HIGH_LOW	Старший байт первым, младший байт последним

Всегда присоединяйте номер элемента к входному-выходному параметру, который передается в In[], например: array[3].

Ниже представлен пример, в котором *Size* = UINT#2, *Initial* = WORD#16#0000, а *OutOrder* = _LOW_HIGH.

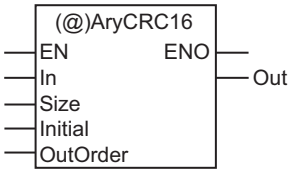


Меры предосторожности для обеспечения надлежащей эксплуатации

- Если значение *Size* равно 0, то значение *Out* равно WORD#16#0.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *OutOrder* находится за пределами допустимого диапазона.
  b) Значение *Size* приводит к выходу за область массива In[].

AryCRC16

Команда AryCRC16 вычисляет значение CRC-16, используя метод MODBUS.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AryCRC16	Расчет CRC-16 массива	FUN		Out:=AryCRC16(In, Size, Initial, OutOrder);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In[] (массив)	Массив для обработки	Вход	Массив для обработки	Зависит от типа данных.	---	*1
Size	Количество элементов для обработки		Количество элементов массива In[]			1
Initial	Начальное значение		Начальное значение CRC-16			16#FFFF
OutOrder	Порядок байтов		Порядок обработки байтов в *In*	_LOW_HIGH, _HIGH_LOW		_LOW_HIGH
Out	Значение CRC-16	Выход	Значение CRC-16	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[] (массив)		OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
Size							OK													
Initial			OK																	
OutOrder	Сведения о перечислителях перечислимого типа _eBYTE_ORDER см. в разделе *Функция* на стр. 2-622.																			
Out			OK																	

Функция

Команда AryCRC16 вычисляет значение CRC-16 для *Size* элементов массива In[] (массив для обработки), начиная с элемента In[0]. Используется метод MODBUS.

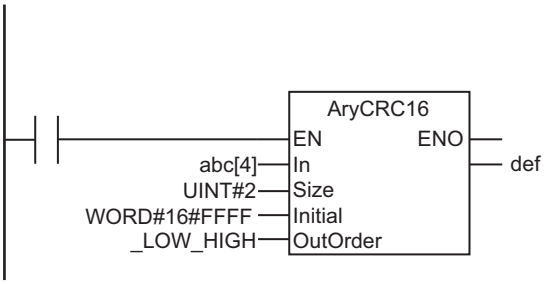
В параметре *Initial* следует указать начальное значение для расчета значения CRC-16. В параметре *OutOrder* задаётся порядок обработки байтов.
Для параметра *OutOrder* используется перечислимый тип данных _eBYTE_ORDER. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_LOW_HIGH	Младший байт первым, старший байт последним
_HIGH_LOW	Старший байт первым, младший байт последним

Всегда присоединяйте номер элемента к входному параметру, который передается в In[], например: array[3].

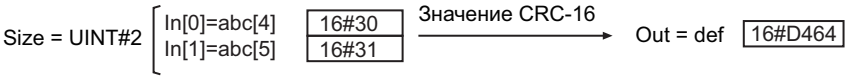
Ниже представлен пример, в котором *Size* = UINT#2, *Initial* = WORD#16#FFFF, а *OutOrder* = _LOW_HIGH.

LD



ST

```
def:=AryCRC16(abc[4], UINT#2,
              WORD#16#FFFF,
              _LOW_HIGH);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

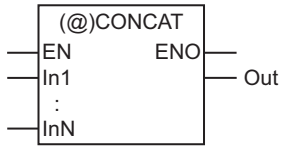
- Если значение *Size* равно 0, то значение *Out* равно WORD#16#0.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *OutOrder* находится за пределами допустимого диапазона.
  b) Значение *Size* приводит к выходу за область массива In[].

Команды для обработки текстовых строк

Команда	Имя	Стр.
CONCAT	Конкатенация строк	стр. 2-626
LEFT и RIGHT	Получить строку слева/Получить строку справа	стр. 2-628
MID	Получить строку в указанной позиции	стр. 2-631
FIND	Поиск строки	стр. 2-633
LEN	Длина строки	стр. 2-635
REPLACE	Замена строки	стр. 2-637
DELETE	Удаление строки	стр. 2-639
INSERT	Вставка строки	стр. 2-641
GetByteLen	Получить длину в байтах	стр. 2-643
ClearString	Очистка строки	стр. 2-645
ToUCase и ToLCase	Перевод в верхний регистр/Перевод в нижний регистр	стр. 2-647
TrimL и TrimR	Обрезка строки слева/Обрезка строки справа	стр. 2-649
AddDelimiter	Поместить в текстовую строку с разделителями	стр. 2-651
SubDelimiter	Получить текстовые строки без разделителей	стр. 2-664

CONCAT

Команда CONCAT объединяет от двух до пяти текстовых строк.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CONCAT	Конкатенация строк	FUN		Out:=CONCAT(In1,⋯, InN);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1...InN	Строки для объединения	Вход	Текстовые строки для объединения, где N = 2...5	Зависит от типа данных.	---	" *1
Out	Результат объединения	Выход	Текстовая строка, полученная в результате объединения	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, подключаемый ко входу *InN*, значение по умолчанию применено не будет и произойдет ошибка сборки.
Например, если N = 3, а входные параметры, которые подключаются к *In1* и *In2*, опущены, то применяются значения по умолчанию. Но если будет опущен входной параметр, который подключается к *In3*, то произойдет ошибка сборки.

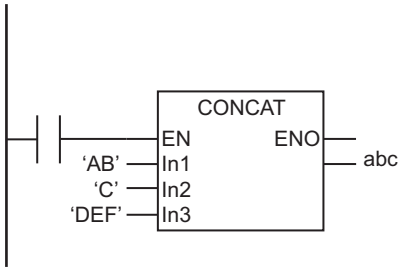
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1...InN																				OK
Out																				OK

Функция

Команда CONCAT объединяет от 2 до 5 текстовых строк *In1...InN* (строки для объединения) в соответствующем порядке. В конец результирующей строки добавляется пустой символ (NULL).

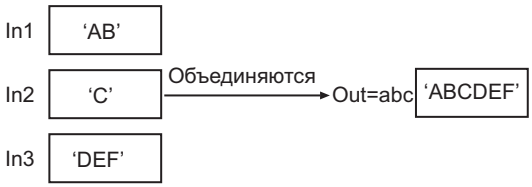
Ниже приведен пример, в котором *In1* = «AB», *In2* = «C», а *In3* = «DEF». Значение переменной *abc* будет равно «ABCDEF».

LD



ST

```
abc:=CONCAT('AB', 'C', 'DEF');
```



Меры предосторожности для обеспечения надлежащей эксплуатации

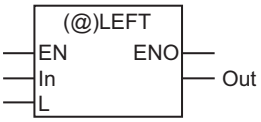
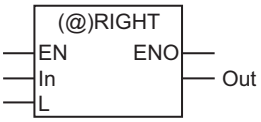
В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Длина объединенной строки символов превышает 1986 байт.

LEFT и RIGHT

Эти команды извлекают из текстовой строки подстроку с указанным количеством символов.

LEFT : Извлекает символы в начале (слева) текстовой строки.
RIGHT : Извлекает символы в конце (справа) текстовой строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LEFT	Получить строку слева	FUN		Out:=LEFT(In, L);
RIGHT	Получить строку справа	FUN		Out:=RIGHT(In, L);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Исходная строка	Вход	Текстовая строка, из которой извлекаются символы	Зависит от типа данных.	---	"
L	Количество символов		Количество извлекаемых символов	0...1985		1
Out	Результат извлечения	Выход	Извлеченная текстовая строка	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
L							OK													
Out																				OK

Функция

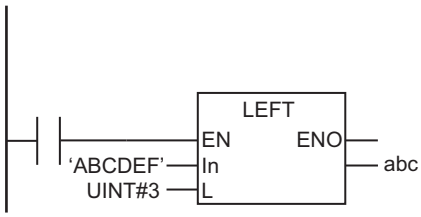
Эти команды извлекают из строки *In* (исходная строка) текстовую строку, количество символов в которой задано параметром *L*. В конец текстовой строки *Out* (результат извлечения) добавляется пустой символ (NULL).

LEFT

Извлекает символы, расположенные слева (в начале) текстовой строки *In*.

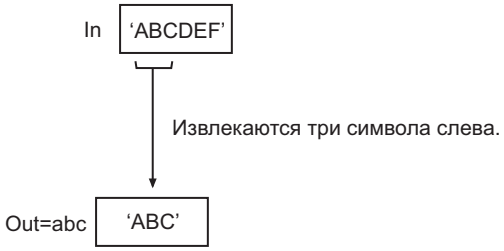
Ниже приведен пример, в котором *In* = «ABCDEF», а *L* = UINT#3. Значение переменной *abc* будет равно «ABC».

LD



ST

```
abc:=LEFT('ABCDEF', UINT#3);
```

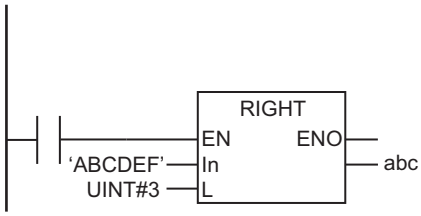


RIGHT

Извлекает символы, расположенные справа (в конце) текстовой строки *In*.

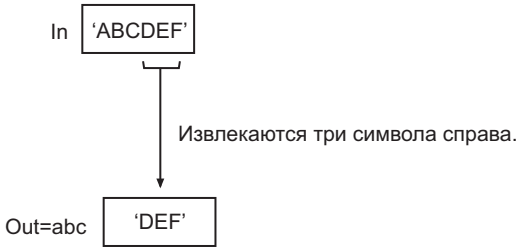
Ниже приведен пример, в котором *In* = «ABCDEF», а *L* = UINT#3. Значение переменной *abc* будет равно «DEF».

LD



ST

```
abc:=RIGHT('ABCDEF', UINT#3);
```

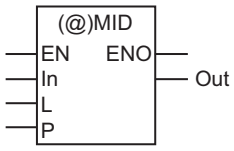


Меры предосторожности для обеспечения надлежащей эксплуатации

- Если значение *L* превышает количество символов в строке *In* или находится в допустимом диапазоне, ошибки не возникает. В этом случае в *Out* копируются все символы строки *In*.
- Если значение *L* равно 0, ошибки не возникает. В этом случае в *Out* записывается только пустой символ (NULL).
- Каждый многобайтовый символ засчитывается как один символ.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Содержимое *In* приводит к ошибке кода символа.

MID

Команда MID извлекает из текстовой строки подстроку с указанным количеством символов, расположенную в указанной позиции.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MID	Получить строку в указанной позиции	FUN		Out:=MID(In, L, P);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Исходная строка	Вход	Текстовая строка, из которой извлекаются символы	Зависит от типа данных.	---	"
L	Количество символов		Количество извлекаемых символов	0...1985		1
P	Первый символ		Первый символ для извлечения	1...1985		
Out	Результат извлечения	Выход	Извлеченная текстовая строка	Зависит от типа данных.	---	---

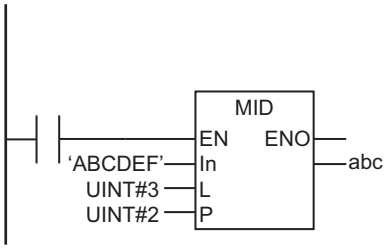
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
L							OK													
P							OK													
Out																				OK

Функция

Команда MID извлекает из строки *In* (исходная строка) текстовую строку, количество символов в которой задано параметром *L*. Первый извлекаемый символ задается параметром *P*. В конец текстовой строки *Out* (результат извлечения) добавляется пустой символ (NULL).

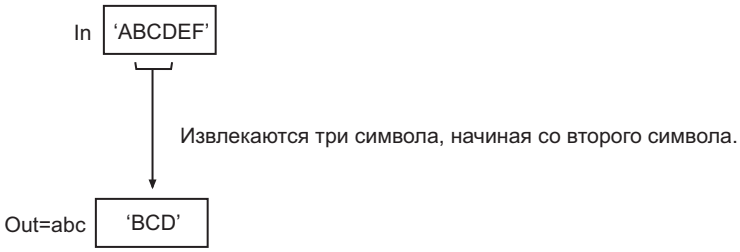
Ниже приведен пример, в котором *In* = «ABCDEF», *L* = UINT#3, а *P* = UINT#2. Значение переменной *abc* будет равно «BCD».

LD



ST

```
abc:=MID('ABCDEF', UINT#3, UINT#2);
```

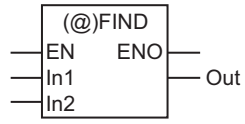


Меры предосторожности для обеспечения надлежащей эксплуатации

- Если значение *L* равно 0, ошибки не возникает. В этом случае в *Out* записывается только пустой символ (NULL).
- Каждый многобайтовый символ засчитывается как один символ.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Содержимое *In* приводит к ошибке кода символа.
  b) Строка *In* не содержит *L* символов после позиции, указанной параметром *P*.
  c) Значение *P* равно 0.

FIND

Команда FIND определяет позицию указанной подстроки в текстовой строке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FIND	Поиск строки	FUN		Out:=FIND(In1, In2);

Переменные

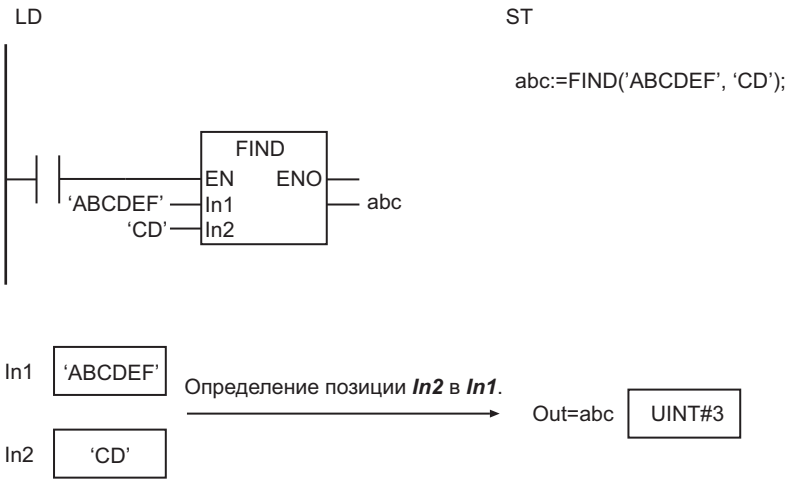
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Строка для поиска	Вход	Текстовая строка, в которой производится поиск	Зависит от типа данных.	---	''
In2	Ключ поиска		Искомая текстовая строка			
Out	Результат поиска	Выход	Результат поиска	0...1985	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																				ОК
In2																				ОК
Out							ОК													

Функция

Команда FIND производит поиск текстовой строки *In2* (ключ поиска) в текстовой строке *In1* (строка для поиска). В переменную *Out* (результат поиска) записывается позиция расположения строки *In2*, отсчитываемая от начала строки *In1*.
Если строка *In2* не найдена в строке *In1*, то *Out* = 0.

Ниже приведен пример для случая, когда *In1* = «ABCDEF», а *In2* = «CD». Значение переменной *abc* будет равно UINT#3.



Меры предосторожности для обеспечения надлежащей эксплуатации

- Количество символов в строке *In2* должно быть меньше количества символов в строке *In1*. В противном случае *Out* будет равно 0.
- Если строка *In1* содержит не одну, а несколько строк *In2*, в *Out* записывается позиция первой найденной строки *In2* от начала строки *In1*.
- Если обе строки, *In1* и *In2*, содержат только пустые символы (NULL), значение *Out* будет равно 1.
- Каждый многобайтовый символ засчитывается как один символ.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  а) Содержимое *In1* или *In2* приводит к ошибке кода символа.

LEN

Команда LEN определяет количество символов в текстовой строке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LEN	Длина строки	FUN	(@)LEN EN ENO In Out	Out:=LEN(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Строка для определения длины	Вход	Текстовая строка для определения длины	Зависит от типа данных.	---	''
Out	Результат определения	Выход	Результат определения длины	0...1985	---	---

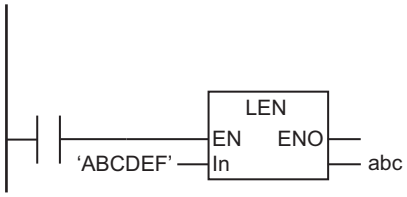
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Out							OK													

Функция

Команда LEN определяет количество символов в строке *In* (строка для определения длины). Пустой символ (NULL) в конце строки *In* в количестве символов не учитывается.

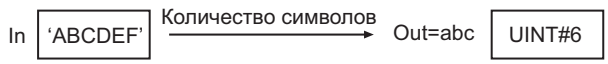
В приведенном ниже примере *In* = «ABCDEF». Значение переменной *abc* будет равно UINT#6.

LD



ST

```
abc:=LEN('ABCDEF');
```

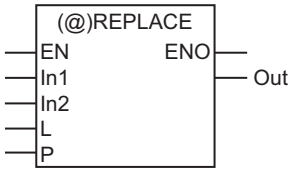


Меры предосторожности для обеспечения надлежащей эксплуатации

- Каждый многобайтовый символ засчитывается как один символ.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  а) Содержимое *In* приводит к ошибке кода символа.

REPLACE

Команда REPLACE заменяет часть текстовой строки другой текстовой строкой.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
REPLACE	Замена строки	FUN		Out:=REPLACE(In1, In2, L, P);

Переменные

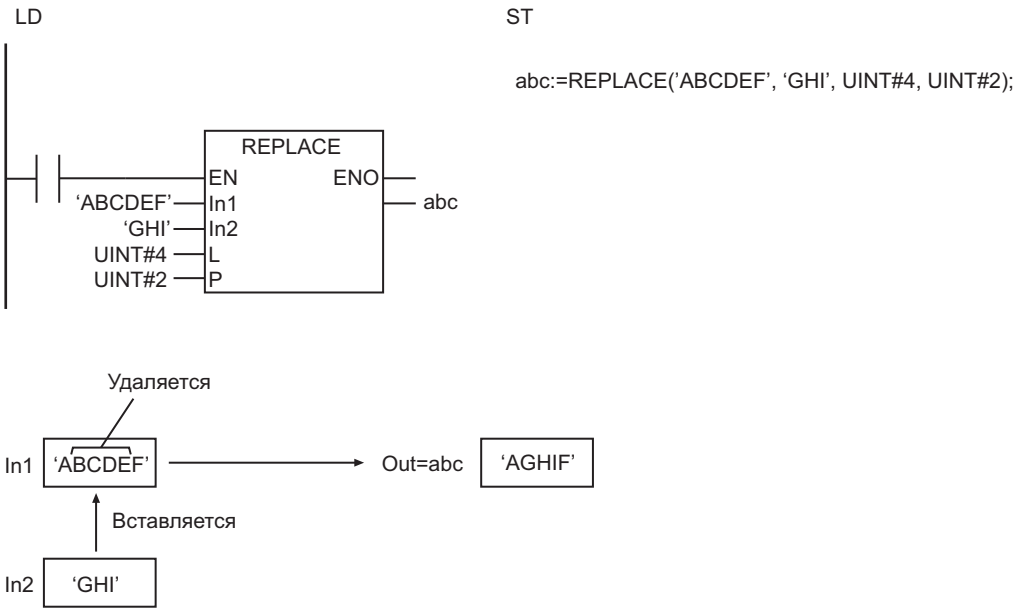
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Строка для замены	Вход	Текстовая строка, в которой производится замена	Зависит от типа данных.	---	"
In2	Вставляемая строка		Вставляемая текстовая строка			
L	Количество символов		Количество удаляемых символов	0...1985		1
P	Позиция начала замены		Позиция начала замены	1...1985		
Out	Результат замены	Выход	Текстовая строка после замены	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																				OK
In2																				OK
L							OK													
P							OK													
Out																				OK

Функция

Команда REPLACE заменяет часть строки *In1* (строка для замены) строкой *In2* (вставляемая строка).
Сначала из строки *In1* удаляется *L* символов, начиная с позиции *P*. Затем на место удаленных символов вставляется строка *In2*. В конец текстовой строки *Out* (результат замены) добавляется пустой символ (NULL).

Ниже приведен пример, в котором *In1* = «ABCDEF», *In2* = «GHI», *P* = UINT#2, а *L* = UINT#4. Значение переменной *abc* будет равно «AGHIF».

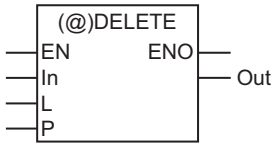


Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *L* = 0, ошибки не произойдет. В этом случае в *Out* будут вставлены все символы *In1*.
- Если *In2* = 0, то из строки *In1* будет удалено *L* символов в позиции *P*.
- Каждый многобайтовый символ засчитывается как один символ.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Содержимое *In1* приводит к ошибке кода символа.
  b) Строка *In1* не содержит *L* символов после позиции, указанной параметром *P*.
  c) Значение *P* равно 0.
  d) Длина строки символов после замены превышает 1986 байт.

DELETE

Команда DELETE удаляет всю текстовую строку или ее часть.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DELETE	Удаление строки	FUN		Out:=DELETE(In, L, P);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Строка для удаления	Вход	Текстовая строка для удаления	Зависит от типа данных.	---	"
L	Количество символов		Количество удаляемых символов	0...1985		1
P	Позиция начала удаления		Позиция начала удаления	1...1985		
Out	Результат удаления	Выход	Текстовая строка после удаления	Зависит от типа данных.	---	---

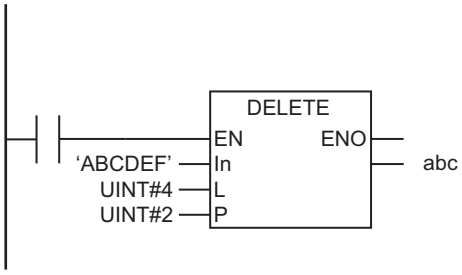
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
L							OK													
P							OK													
Out																				OK

Функция

Команда DELETE удаляет *L* символов из строки *In*, начиная с позиции *P*. В конец текстовой строки *Out* (результат удаления) добавляется пустой символ (NULL).

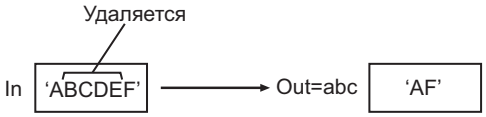
Ниже приведен пример, в котором *In* = «ABCDEF», *L* = UINT#4, а *P* = UINT#2. Значение переменной *abc* будет равно «AF».

LD



ST

```
abc:=DELETE('ABCDEF', UINT#4, UINT#2);
```

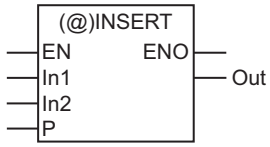


Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *L* = 0, ошибки не произойдет. В этом случае в *Out* будут вставлены все символы *In*.
- Каждый многобайтовый символ засчитывается как один символ.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Содержимое *In* приводит к ошибке кода символа.
  b) Строка *In* не содержит *L* символов после позиции, указанной параметром *P*.
  c) Значение *P* равно 0.

INSERT

Команда INSERT вставляет текстовую строку в другую текстовую строку.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
INSERT	Вставка строки	FUN		Out:=INSERT(In1, In2, P);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Исходная строка	Вход	Текстовая строка, в которую вставляется строка	Зависит от типа данных.	---	"
In2	Вставляемая строка		Вставляемая текстовая строка			
P	Позиция начала вставки		Позиция начала вставки	0...1985		0
Out	Результат вставки	Выход	Текстовая строка после вставки	Зависит от типа данных.	---	---

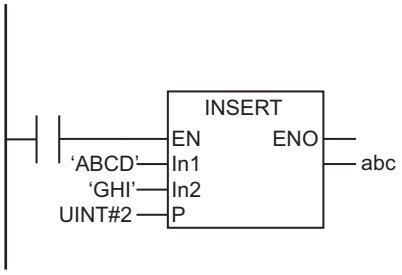
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																				OK
In2																				OK
P							OK													
Out																				OK

Функция

Команда INSERT вставляет строку *In2* (вставляемая строка) в позицию *P* (позиция начала вставки) строки *In1* (исходная строка). В конец текстовой строки *Out* (результат вставки) добавляется пустой символ (NULL).

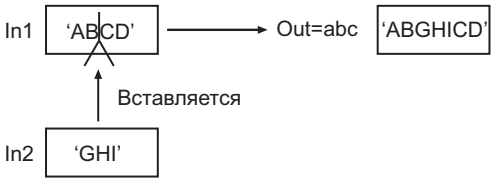
Ниже приведен пример, в котором *In1* = «ABCD», *In2* = «GHI», а *P* = UINT#2. Значение переменной *abc* будет равно «ABGHICD».

LD



ST

```
abc:=INSERT('ABCD', 'GHI', UINT#2);
```



Дополнительная информация

Если *P* = 0, *In1* вставляется в конец *In2*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Каждый многобайтовый символ засчитывается как один символ.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Содержимое *In1* приводит к ошибке кода символа.
  b) Значение *P* больше, чем количество символов в *In1*.
  c) Длина строки символов после вставки превышает 1986 байт.

GetByteLen

Команда GetByteLen подсчитывает количество байтов в текстовой строке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetByteLen	Получить длину в байтах	FUN	(@)GetByteLen EN ENO In Out	Out:=GetByteLen(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Строка для подсчета	Вход	Текстовая строка для подсчета количества байтов	Зависит от типа данных.	---	''
Out	Количество байтов	Выход	Количество байтов	0...1985	Байты	---

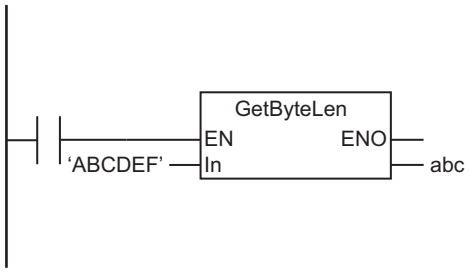
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Out							OK													

Функция

Команда GetByteLen подсчитывает количество байтов в строке *In* (строка для подсчета). Пустой символ (NULL) в конце текстовой строки в количестве байтов не учитывается.

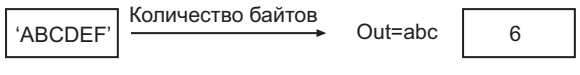
В приведенном ниже примере *In* = «ABCDEF». Значение переменной *abc* будет равно 6.

LD



ST

```
abc:=GetByteLen('ABCDEF');
```

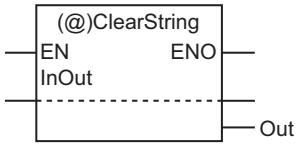


Дополнительная информация

Если строка *In* содержит только символы ASCII, результат будет таким же, как при выполнении команды LEN.

ClearString

Команда ClearString очищает текстовую строку.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ClearString	Очистка строки	FUN		ClearString(InOut);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
InOut	Очищаемая строка	Вход-выход	Очищаемая текстовая строка	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

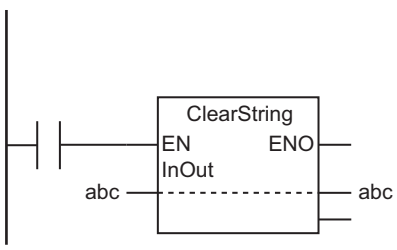
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InOut																				ОК
Out	ОК																			

Функция

Команда ClearString очищает строку *InOut* (очищаемая строка). Вся строка *InOut* заполняется пустыми символами (NULL).

Пример программы представлен на рисунке ниже. Переменная *abc* типа STRING будет содержать только пустые символы (NULL).

LD



ST

```
ClearString(abc);
```

Команда ClearString заполняет всю строку *InOut* пустыми символами (NULL).

Ниже показан пример для случая, когда *abc* является переменной типа STRING, содержащей 5 символов.

Команда ClearString заполняет всю строку ***InOut*** пустыми символами (NULL).

Ниже приведен пример для случая, когда ***abc*** — переменная типа STRING с 5 символами.

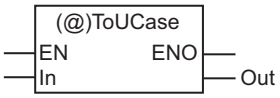
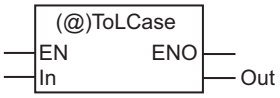
InOut=abc	NULL	NULL	NULL	NULL	NULL

Меры предосторожности для обеспечения надлежащей эксплуатации

При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.

ToUCase и ToLCase

ToUCase : Переводит все однобайтовые буквы в текстовой строке в верхний регистр.
ToLCase : Переводит все однобайтовые буквы в текстовой строке в нижний регистр.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ToUCase	Перевод в верхний регистр	FUN		Out:=ToUCase(In);
ToLCase	Перевод в нижний регистр	FUN		Out:=ToLCase(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные для преобразования	Вход	Текстовая строка для преобразования	Зависит от типа данных.	---	"
Out	Результат преобразования	Выход	Преобразованная текстовая строка	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Out																				OK

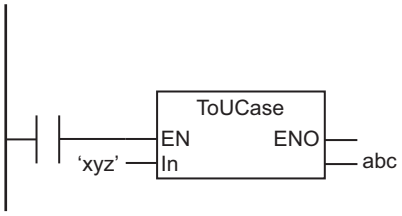
Функция

ToUCase

Команда ToUCase переводит все однобайтовые буквы в строке *In* (данные для преобразования) в верхний регистр.
В конце текстовой строки эта команда помещает пустой символ (NULL). Изменяются только однобайтовые символы.

Ниже показан пример использования команды ToUCase, в котором *In* = «xyz». Значение переменной *abc* будет равно «XYZ».

LD

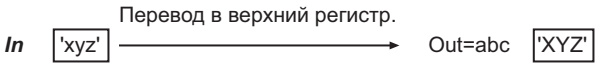


ST

```
abc:=ToUCase('xyz');
```

Команда ToUCase преобразует все однобайтовые буквы в строке *In* в прописные буквы.

Команда ToUCase переводит все однобайтовые буквы в ***In*** в верхний регистр.



ToLCase

Команда ToLCase переводит все однобайтовые буквы в строке *In* в нижний регистр.
В конце текстовой строки эта команда помещает пустой символ (NULL). Изменяются только однобайтовые символы.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Двухбайтовые буквы не преобразуются.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Содержимое *In* приводит к ошибке кода символа.

TrimL и TrimR

TrimL : Удаляет пробел в начале текстовой строки.
TrimR : Удаляет пробел в конце текстовой строки.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TrimL	Обрезка строки слева	FUN		Out:=TrimL(In);
TrimR	Обрезка строки справа	FUN		Out:=TrimR(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Строка для обрезки	Вход	Обрезаемая текстовая строка	Зависит от типа данных.	---	''
Out	Результат обрезки	Выход	Текстовая строка после обрезки	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Out																				OK

Функция

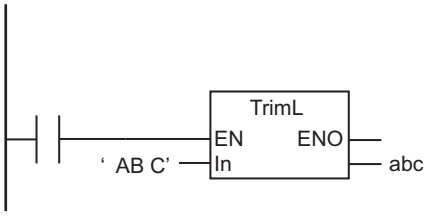
TrimL

Команда TrimL удаляет символы пробела в начале строки *In* (строка для обрезки).
Если в начале текстовой строки нет пробелов, никаких действий не производится.

В конце текстовой строки эта команда помещает пустой символ (NULL). Символами пробела считаются как символы пробела ASCII (16#20), так и двухбайтовые японские символы пробела (16#E38080).

Ниже показан пример использования команды TrimL для случая, когда *In* = « AB C». Значение переменной *abc* будет равно «AB C».

LD

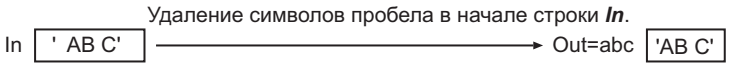


ST

```
abc:=TrimL(’ AB C’);
```

Команда TrimL удаляет символы пробела в начале строки *In*.

Команда TrimL удаляет символы пробела в начале строки ***In***.



TrimR

Команда TrimR удаляет символы пробела в конце строки *In* (строка для обрезки).
Если в конце текстовой строки нет пробелов, никаких действий не производится.

В конце текстовой строки эта команда помещает пустой символ (NULL). Символами пробела считаются как символы пробела ASCII (16#20), так и двухбайтовые японские символы пробела (16#E38080).

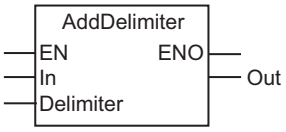
Меры предосторожности для обеспечения надлежащей эксплуатации

В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Содержимое *In* приводит к ошибке кода символа.

AddDelimiter

Команда AddDelimiter преобразует значения всех членов в структуре в текстовую строку с разделителями.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AddDelimiter	Поместить в текстовую строку с разделителями	FUN		Out:=AddDelimiter(In, Delimiter);

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Входная структура	Вход	Структура для преобразования в текстовые строки	Зависит от типа данных членов структуры.	---	*1
Delimiter	Разделитель		Разделитель	_COMMA, _TAB, _SEMICOLON, _SPACE	---	_COMMA
Out	Возвращаемое значение	Выход	Текстовые строки с разделителями	Макс. 1986 байт (1985 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	Структура																			
Delimiter	Сведения о перечислителях перечислимого типа _eDELIMITER см. в разделе *Функция* на стр. 2-652.																			
Out																				OK

Функция

Команда AddDelimiter преобразует каждый член структуры *In* (входная структура) в текстовую строку, начиная с самого первого члена структуры и далее по порядку, а затем объединяет полученные строки с использованием разделителя, заданного параметром *Delimiter*. Объединенная текстовая строка выводится в переменную *Out* (возвращаемое значение). В конец строки *Out* добавляется нулевой символ (NULL).

Для параметра *Delimiter* используется перечислимый тип данных _eDELIMITER. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_COMMA	«,» (запятая)
_TAB	«$T» (табулятор)
_SEMICOLON	«;» (точка с запятой)
_SPACE	« » (пробел)

Значения членов структуры *In* преобразуются в соответствии с их типами данных.

● Значения логического типа

ЛОЖЬ преобразуется в «0», а ИСТИНА — в «1».

● Битовые строки

Битовые строки воспринимаются как шестнадцатеричные числа и преобразуются в текстовые строки с буквенно-цифровыми символами, которые соответствуют этим числам. Префикс шестнадцатеричного числа 16# в текстовую строку не выводится.
Если для значения члена структуры требуется меньше разрядов, чем предусмотрено типом данных члена структуры, старшие разряды будут содержать «0». Другими словами, неиспользуемые разряды заполняются нулями.
Количество символов в текстовой строке зависит от типа данных, что отражено в представленной ниже таблице.

Тип данных члена структуры	Количество символов
BYTE	2 однобайтовых буквенно-цифровых символа
WORD	4 однобайтовых буквенно-цифровых символа
DWORD	8 однобайтовых буквенно-цифровых символов
LWORD	16 однобайтовых буквенно-цифровых символов

Ниже приведено несколько примеров.

Значение члена структуры	Преобразованная текстовая строка
BYTE#16#AB	«AB»
LWORD#16#0123	«0000000000000123»

● Целые числа

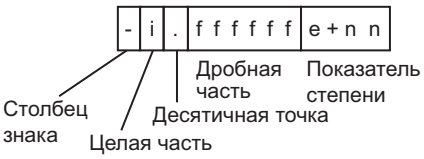
Значение целого числа преобразуется в текстовую строку. Старшие разряды, содержащие 0, в текстовую строку не выводятся. Если значение члена структуры является отрицательным числом, то в начало текстовой строки добавляется знак минуса (-).

Ниже приведено несколько примеров.

Значение члена структуры	Преобразованная текстовая строка
UINT#0012	«12»
LINT#-12	«-12»

● Вещественные числа

Структура текстовой строки, в которую преобразуется значение члена структуры, показана ниже.



Элемент	Описание
Столбец знака	Если значение члена структуры является отрицательным числом, добавляется знак минуса (-). Если значение члена структуры является положительным числом, знак плюса (+) не добавляется.
Целая часть	Целая часть всегда состоит только из одной цифры.
Десятичная точка	Десятичная точка всегда имеется, даже если значение члена структуры не является десятичным числом.
Дробная часть	Если член структуры относится к типу данных REAL, то выводится 6 разрядов. Если член структуры относится к типу данных LREAL, то выводится 14 разрядов.
Показатель степени	Показатель степени всегда имеется. Буква «e» обозначает показатель степени "e". «nn» — это 2 или 3 цифры. Знак «nn» положителен (+), если абсолютное значение члена структура равно 1,0 или больше, и отрицателен (-), если оно меньше 1,0. Если значение члена структуры равно 0, то эта часть положительна (+).

В таблице ниже приведены значения текстовой строки для случаев, когда член структуры содержит бесконечность или нечисловое значение.

Значение члена структуры	Текстовая строка
+ ∞	«inf»
-∞	«-inf»
Нечисловое значение	«nan» или «-nan»

Ниже приведено несколько примеров.

Значение члена структуры	Преобразованная текстовая строка
REAL#3.14e1	«3.140000e+01»
REAL#-123.4567	«-1.234567e+02»
REAL#0	«0.000000e+00»
LREAL#0.00123456789	«1.23456789000000e-03»
LREAL#1.0e308	«1.00000000000000e+308»

● Продолжительность

Структура текстовой строки, в которую преобразуется значение члена структуры, показана ниже.

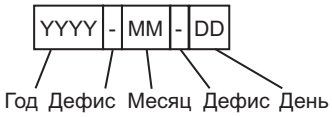
Элемент	Описание
Столбец знака	Если значение члена структуры является отрицательным числом, добавляется знак минуса (-). Если значение члена структуры является положительным числом, знак плюса (+) не добавляется.
Дни	Количество дней указывается всегда. Диапазон допустимых значений: от 0 до 106751. Старшие разряды нулями (0) не заполняются.
Часы	Количество часов указывается всегда в двух разрядах. Диапазон допустимых значений: от 00 до 23.
Минуты	Количество минут указывается всегда в двух разрядах. Диапазон допустимых значений: от 00 до 59.
Секунды	Количество секунд указывается всегда. Значение «DD» указывается всегда в двух разрядах в диапазоне от 00 до 59. Значение «EE» указывается всегда в двух разрядах в диапазоне от 000000000 до 999999999.
«d», «h», «m», «s» и десятичная точка	Выходная строка всегда содержит эти элементы.

Ниже приведено несколько примеров.

Значение члена структуры	Преобразованная текстовая строка
T#-180122000ms	«-2d02h02m02.000000000s»
T#100d2h3m5.678s	«100d02h03m05.678000000s»
T#2h3m5.678s	«0d02h03m05.678000000s»

● Дата

Структура текстовой строки, в которую преобразуется значение члена структуры, показана ниже.

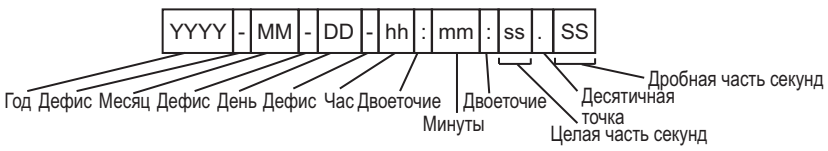


Значение месяца и значение дня преобразуются в две цифры каждое и выводятся в текстовую строку.
Примеры приведены ниже.

Значение члена структуры	Преобразованная текстовая строка
D#2010-1-2	«2010-01-02»

● Дата и время

Структура текстовой строки, в которую преобразуется значение члена структуры, показана ниже.

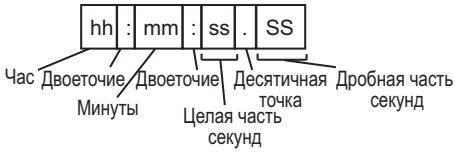


Значения месяца (MM), дня (DD), часов (hh), минут (mm), а также целая часть значения секунд (ss) преобразуются в две цифры каждое и выводятся в текстовую строку. Дробная часть значения секунд (ss) преобразуется в девять цифр и выводится в текстовую строку.
Примеры приведены ниже.

Значение члена структуры	Преобразованная текстовая строка
DT#2004-09-23-12:16:8.12	«2004-09-23-12:16:08.120000000»

● Время суток

Структура текстовой строки, в которую преобразуется значение члена структуры, показана ниже.



Значения часов (hh), минут (mm), а также целая часть значения секунд (ss) преобразуются в две цифры каждое и выводятся в текстовую строку. Дробная часть значения секунд (ss) преобразуется в девять цифр и выводится в текстовую строку.
Примеры приведены ниже.

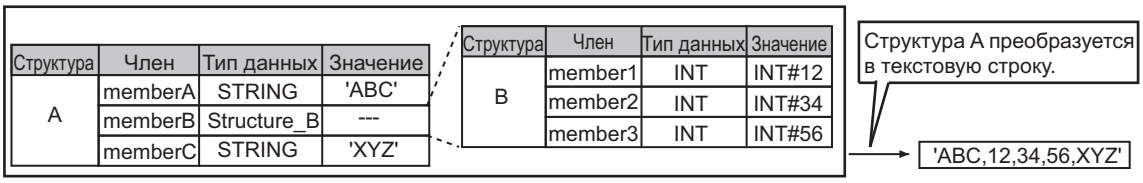
Значение члена структуры	Преобразованная текстовая строка
TOD#2:16:28.12	«02:16:28.120000000»

● Текстовые строки

Текстовая строка выводится без каких-либо изменений. Пустой символ (NULL) в конце текстовой строки в выводимую строку не включается.
Например, если член структуры содержит строку «ABC» с пустым символом (NULL) в конце строки, в выходную текстовую строку будут выведены только символы «ABC», а пустой символ (NULL) выведен не будет.

● Структуры

Значения членов структуры преобразуются по порядку, от начала структуры до нижележащих (вложенных) уровней, данные на которых не являются структурами. Значения членов структуры преобразуются в текстовые строки в соответствии с правилами, которые применяются к их типам данных.
Например, если член структуры A имеет тип данных Structure_B, преобразование выполняется показанным ниже образом. В этом примере в качестве разделителей используются запятые.



● Перечисления

Значение перечисления обрабатывается как данные типа DINT и преобразуется соответствующим образом.

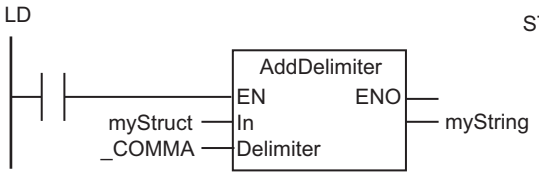
В качестве примера рассмотрим перечисление *Color* с тремя перечислителями: *red*, *yellow* и *green*. Эти перечислители представляют следующие числовые значения: *red* = 1, *yellow* = 2, *green* = 3. Если значение члена перечисления *Color* = *yellow*, то будет выведена текстовая строка «2».

● Массивы

Текстовые строки, соответствующие элементам массива, разделяются разделителем. Значение каждого элемента преобразуется в соответствии с правилами преобразования, которые применяются к типу данных массива. Преобразуются только одномерные массивы.
В качестве примера рассмотрим массив myArray[0..2] типа INT. Если myArray[0] = INT#225, myArray[1] = INT#-128, myArray[2] = INT#0, а в качестве разделителя используется запятая, выходная текстовая строка будет выглядеть следующим образом: «225,-128,0».

Пример записи

Представленный ниже пример демонстрирует, как структура *myStruct* преобразуется в текстовую строку *myString*. В качестве разделителя используется «,» (запятая).




```
ST
myString:=AddDelimiter(myStruct,_COMMA);
```

Структура	Член	Тип данных	Значение
myStruct	member1	STRING	'XYZ'
	member2	INT	INT#12
	member3	WORD	WORD#16#00AB
	member4	REAL	REAL#3.14e0

Преобразование в текстовую строку.

Out = myString 'XYZ,12,00AB,3.140000e+00'

Дополнительная информация

- Данную команду можно использовать в сочетании с командой *FilePuts* на стр. 2-1585 для простой записи значений в указанные файлы CSV на карте памяти SD. Пример применения команды см. в разделе *Пример программы* на стр. 2-657.
- Для чтения текстовых строк, преобразованных с помощью команды AddDelimiter, и их вывода в качестве значений членов структуры можно использовать команду *SubDelimiter* на стр. 2-664.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Не следует включать разделитель в значение члена структуры в *In*. Если значение члена структуры в *In* будет содержать знак разделителя, команда SubDelimiter не сможет корректно преобразовать текстовую строку в значения членов структуры.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Размер результирующей текстовой строки превышает 1986 байт, включая конечный пустой символ (NULL).

b) Член структуры в *In* является массивом с несколькими измерениями.
c) Член структуры в *In* является объединением.

Пример программы

Структура *myStruct* содержит десять членов, которые являются переменными типа SINT.
В данном примере содержимое массива myArray[0..99], элементы которого относятся к структурному типу myStruct, сохраняется в виде 100 строк в файл с именем «ABC.csv» (файл в формате CSV) на карте памяти SD.
Каждая строка файла содержит значения членов элемента массива, преобразованных в 10 текстовых строк. Между этими значениями вставляются запятые. В конец каждой строки файла добавляется код CR + LF.

Обработка выполняется в следующем порядке:

1 Используется команда FileOpen для открытия файла «ABC.csv».

2 Используется команда AddDelimiter для преобразования элемента myArray[] для одной строки файла и вывода результатов в переменную *Temp* типа STRING.

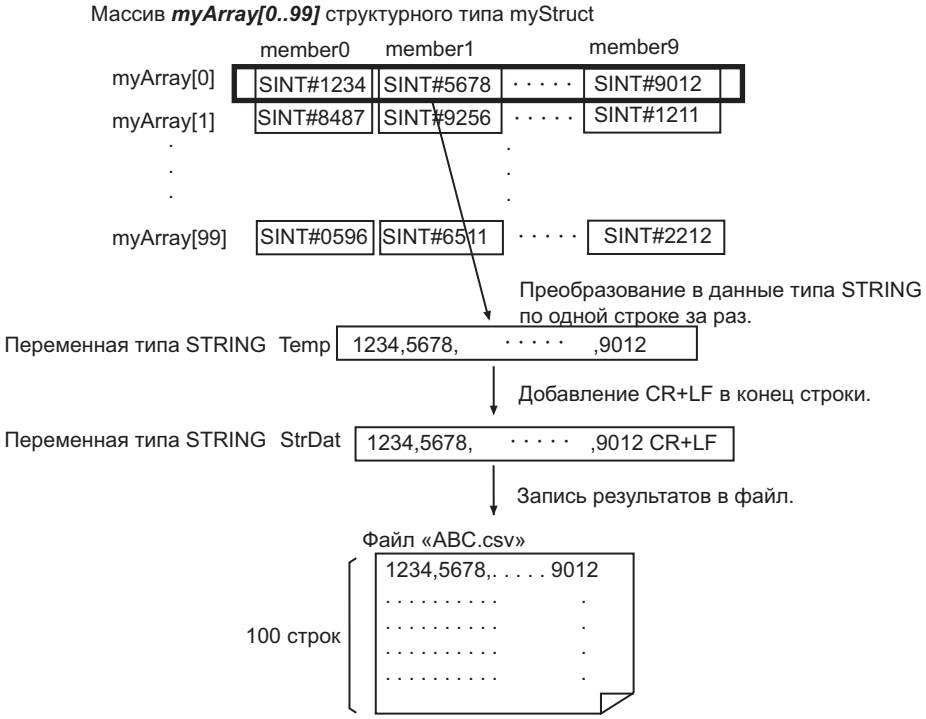
3 Используется команда CONCAT для объединения *Temp* и CR+LF, и результат сохраняется в переменную *StrDat* типа STRING.

4 Содержимое *StrDat* записывается в файл.

5 Шаги 2–4 повторяются для 100 строк файла.

6 Используется команда FileClose для закрытия файла.

Структура	Член	Тип данных
myStruct	member0	SINT
	member1	SINT
	⋮	
	member9	SINT



Определение типов данных

Имя	Тип данных	Комментарий
myStruct	STRUCT	Структура
member0	SINT	Член перечисления
member1	SINT	Член перечисления
member2	SINT	Член перечисления
member3	SINT	Член перечисления
member4	SINT	Член перечисления
member5	SINT	Член перечисления
member6	SINT	Член перечисления
member7	SINT	Член перечисления
member8	SINT	Член перечисления
member9	SINT	Член перечисления

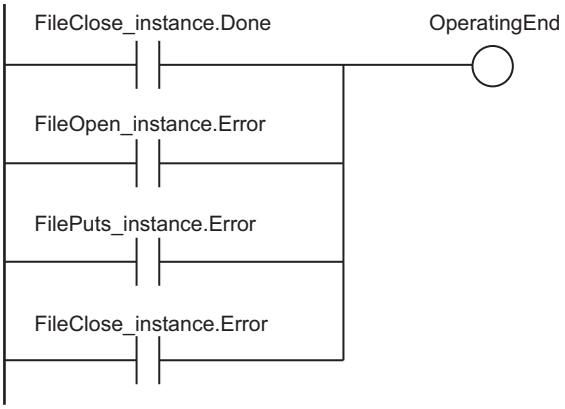
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	Ложь	Обработка завершена
	Trigger	BOOL	Ложь	Условие выполнения

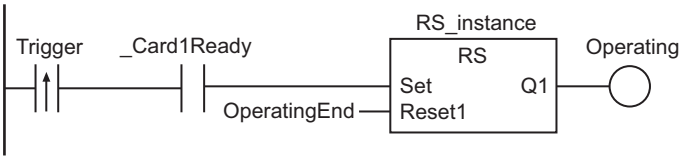
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Operating	BOOL	Ложь	Обработка
	Index	INT	0	Указатель
	Fid	DWORD	16#0	ID файла
	StrDat	STRING[256]	''	Данные текстовой строки
	myArray	ARRAY[0..99] OF myStruct	[100((member0:=0,member1:=0,member2:=0,member3:=0,member4:=0,member5:=0,member6:=0,member7:=0,member8:=0,member9:=0))]	Числовые значения
	Temp	STRING[256]	''	Временные данные
	RS_instance	RS		
	FileOpen_instance	FileOpen		
	FilePuts_instance	FilePuts		
	FileClose_instance	FileClose		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

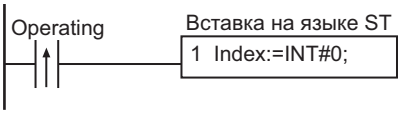
Определение, завершено ли выполнение команды для карты памяти SD.



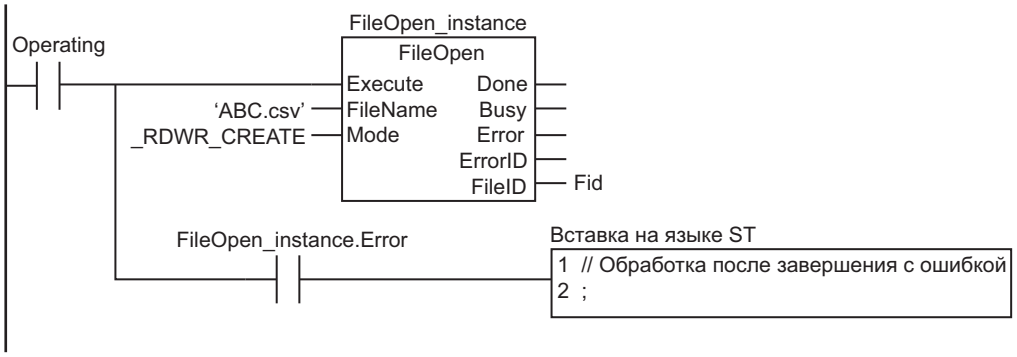
Прием условия выполнения.



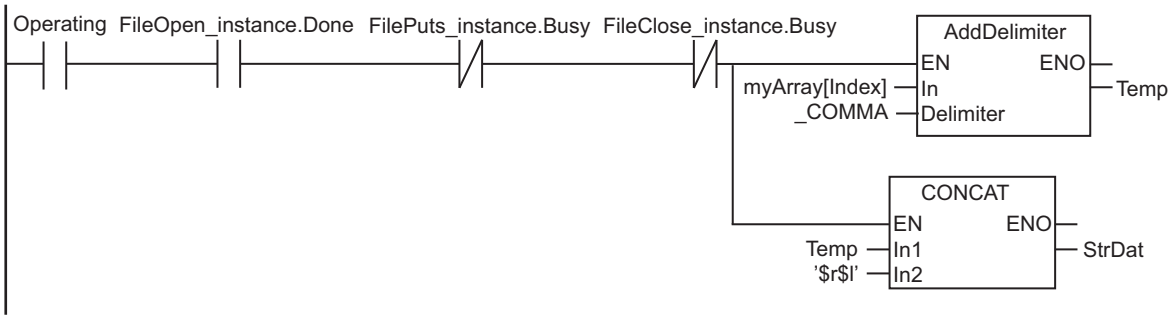
Инициализация индекса строки.



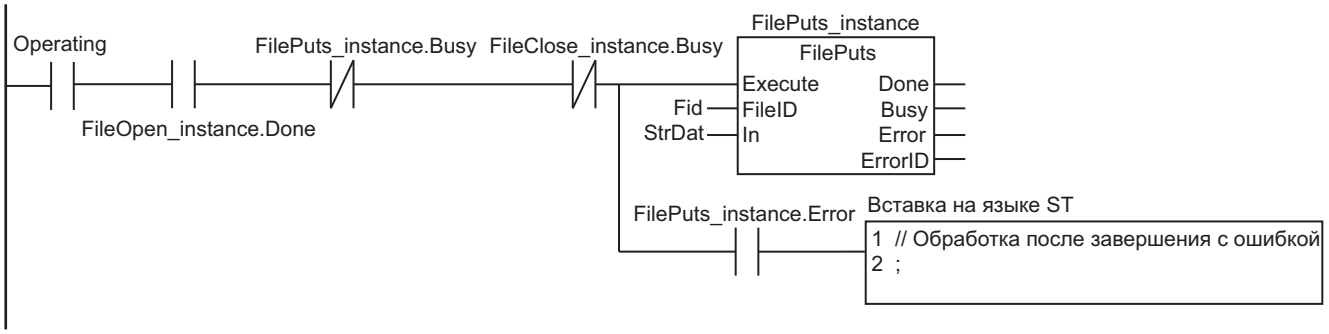
Выполнение команды FileOpen.



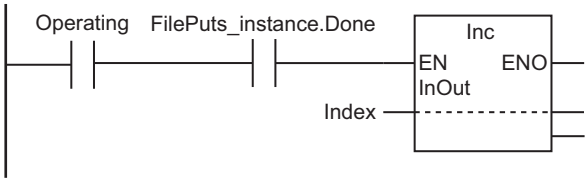
Создание текстовой строки для одной строки.



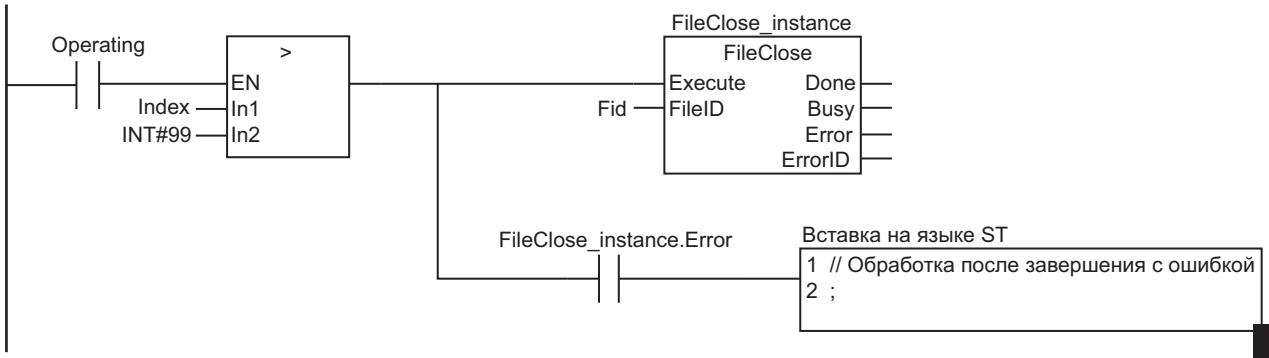
Запись текстовой строки для одной строки в файл.



Увеличение индекса строки на 1.



Выполнить команду FileClose после того, как будет записано 100 строк.



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	Ложь	Условие выполнения
	LastTrigger	BOOL	Ложь	Значение *Trigger* в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	Ложь	Обработка началась
	Operating	BOOL	Ложь	Обработка
	Stage	INT	0	Смена этапа
	Index	INT	0	Index
	Fid	DWORD	16#0	ID файла
	StrDat	STRING[256]	''	Данные текстовой строки
	myArray	ARRAY[0..99] OF myStruct	[100((Member0:=0,member1:=0,member2:=0,member3:=0,member4:=0,member5:=0,member6:=0,member7:=0,member8:=0,member9:=0))]	Числовые значения
	Temp	STRING[256]	''	Временные данные
	FileOpen_instance	FileOpen		
	FilePuts_instance	FilePuts		
	FileClose_instance	FileClose		

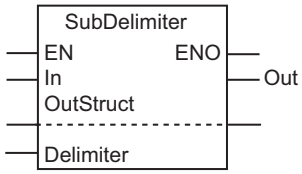
Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
  OperatingStart:=TRUE;
  Operating :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация экземпляра.
IF (OperatingStart=TRUE) THEN
  FileOpen_instance(Execute:=FALSE);
  FilePuts_instance(Execute:=FALSE);
  FileClose_instance(Execute:=FALSE);
  Stage :=INT#1;
  Index :=INT#0; // Инициализация индекса строки.
  OperatingStart:=FALSE;
END_IF;

// Выполнение команды.
IF (Operating=TRUE) THEN
  CASE Stage OF
  1 : // Открытие файла.
    FileOpen_instance(
      Execute :=TRUE,
      FileName:='ABC.csv', // Имя файла
      Mode :=_RDWR_CREATE, // Чтение файла
      FileID =>Fid); // ID файла

    IF (FileOpen_instance.Done=TRUE) THEN
      Stage:=INT#2; // Нормальное завершение
    END_IF;

    IF (FileOpen_instance.Error=TRUE) THEN
      Stage:=INT#99; // Завершение с ошибкой
    END_IF;

  2 : // Создание текстовой строки для одной строки файла.
    StrDat:='';

    Temp :=AddDelimiter(myArray[Index],_COMMA);
    StrDat:=CONCAT(In1:=Temp, In2:='$r$l');

    Stage:=INT#3;

  3 : // Запись текстовой строки.
    FilePuts_instance(
      Execute:=TRUE,
      FileID :=Fid,
      In :=StrDat);

    IF (FilePuts_instance.Done=TRUE) THEN
```

```
      Index:=Index+INT#1;

      IF (Index>INT#99) THEN // Если записано 100 строк
        Stage:=INT#4;
      ELSE
        FilePuts_instance(Execute:=FALSE);
        Stage:=INT#2;
      END_IF;
    END_IF;

    IF (FilePuts_instance.Error=TRUE) THEN
      Stage:=INT#99; // Завершение с ошибкой
    END_IF;

  4 : // Закрытие файла.
    FileClose_instance(
      Execute:=TRUE,
      FileID :=Fid); // Идентификатор файла

    IF (FileClose_instance.Done=TRUE) THEN
      Operating:=FALSE; // Нормальное завершение
    END_IF;

    IF (FileClose_instance.Error=TRUE) THEN
      Stage:=INT#99; // Завершение с ошибкой
    END_IF;

  99 : // Обработка после завершения с ошибкой
    Operating:=FALSE;
  END_CASE;
END_IF;
```

SubDelimiter

Команда SubDelimiter считывает разделенные части текстовой строки и сохраняет их как значения членов структуры.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SubDelimiter	Получить текстовые строки без разделителей	FUN		Out:=SubDelimiter(In, OutStruct, Delimiter);

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Входная тестовая строка	Вход	Текстовая строка с разделенными частями для преобразования в значения членов структуры	Макс. 1986 байт (1985 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	"
Delimiter	Разделитель		Разделитель	_COMMA, _TAB, _SEMICOLON, _SPACE	---	_COMMA
OutStruct	Сохраняемая структура	Вход-выход	Структура для хранения результатов преобразования данных	Макс. 8192 байт	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																				OK
Delimiter	Сведения о перечислителях перечислимого типа _eDELIMITER см. в разделе *Функция* на стр. 2-665.																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
OutStruct	Структура																			
Out	ОК																			

Функция

Команда SubDelimiter преобразует переданные в *In* (входная тестовая строка) текстовые строки, разделенные разделителем *Delimiter*, в значения членов структуры *OutStruct* (сохраняемая структура) и присваивает каждое преобразованное значение соответствующему члену структуры.

Для параметра *Delimiter* используется перечислимый тип данных _eDELIMITER. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_COMMA	«,» (запятая)
_TAB	«$T» (табулятор)
_SEMICOLON	«;» (точка с запятой)
_SPACE	« » (пробел)

Если количество разделенных текстовых строк в *In* превышает количество членов в *OutStruct*, остальные строковые данные игнорируются.
Если количество разделенных текстовых строк в *In* меньше количества членов в *OutStruct*, значения остальных членов структуры не изменяются.
Если некоторый член структуры *OutStruct* сам является структурой и в *In* недостаточно данных для всех членов структуры, данные все равно сохраняются, насколько это возможно.
Если некоторый член структуры *OutStruct* является массивом и в *In* недостаточно данных для всех элементов массива, данные все равно сохраняются, насколько это возможно.

Разделенные данные в переменной *In* должны быть данными типа STRING. Строковые данные (STRING) преобразуются в соответствии с типами данных членов структуры *OutStruct*, как описано ниже.

● Значения логического типа

Если значение типа STRING равно «FALSE» или «0», оно преобразуется в ЛОЖЬ. Если значение типа STRING равно «TRUE» или «1», оно преобразуется в ИСТИНА.
Ниже приведены исключения из этого правила.

- Любые идущие подряд символы «0» перед «0» или «1» игнорируются.
- Регистр букв в словах «FALSE» и «TRUE» не играет роли.

Если значение типа STRING не равно «FALSE», «TRUE», «0» или «1», то преобразование невозможно.

● Битовые строки

Применяются те же правила преобразования, что и для команды *STRING_TO_** (Группа преобразования текстовых строк в битовые строки)* на стр. 2-347.

Преобразование невозможно, если данные не выражают шестнадцатеричное число.

● Целые числа

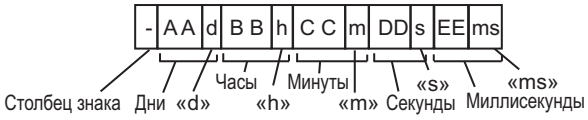
Применяются те же правила преобразования, что и для команды *STRING_TO_** (Группа преобразования текстовых строк в целые числа)* на стр. 2-344.
Преобразование невозможно, если данные не выражают целое число.

● Вещественные числа

Применяются те же правила преобразования, что и для команды *STRING_TO_** (Группа преобразования текстовых строк в вещественные числа)* на стр. 2-350.
Преобразование невозможно, если данные не выражают вещественное число.

● Продолжительность

Ниже показана структура, которую должны иметь данные для преобразования в значение продолжительности.



Элемент	Описание
Столбец знака	Если имеется знак плюса (+) или столбец знака отсутствует, значение члена структуры будет положительным. Если имеется знак минуса (-), значение члена структуры будет отрицательным.
Дни	Значение AA усекается до 11-го разряда после запятой.
Часы	Значение BB усекается до 11-го разряда после запятой.
Минуты	Значение CC усекается до 10-го разряда после запятой.
Секунды	Значение DD усекается до 9-го разряда после запятой.
Миллисекунды	Значение EE усекается до 6-го разряда после запятой.

Примечание1. Знаки пробела (« ») перед столбцом знака, а также перед количеством дней, часов, минут, секунд или миллисекунд игнорируются.

Примечание2. Если какие-либо символы в значениях AA, BB, CC, DD или EE разделены одним символом подчеркивания («_»), этот символ подчеркивания игнорируется.

Примечание3. Даже если количество дней, часов, минут, секунд или миллисекунд является вещественным числом с десятичной точкой («.»), данные все равно могут быть преобразованы.

Примечание4. Если в данные включены дни, часы, минуты, секунды или миллисекунды, преобразование возможно, даже если другие элементы опущены.

Примечание5. Даже если перед значением количества дней, часов, минут, секунд или миллисекунд имеется «0», данные все равно могут быть преобразованы.

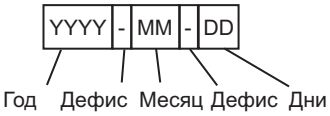
Преобразование невозможно в указанных ниже случаях.

- Структура данных отличается от указанной выше.
- Имеется знак подчеркивания («_») между столбцом знака и количеством дней.
- Данные содержат несколько идущих подряд точек («.») или знаков подчеркивания («_»).

Например, значение строкового типа «-0.5d48h0.123456789ms» будет преобразовано в значение члена структуры T#-2d12h0m0s0.123456ms(T#-216000000.123456ms).

● Дата

Ниже показана структура, которую должны иметь данные для преобразования в значение даты.



Ниже приведены исключения из этого правила.

- Знаки пробела (« ») перед значением года, месяца или дня игнорируются.
- Если какие-либо символы в значении года, месяца или дня разделены одним символом подчеркивания («_»), этот символ подчеркивания игнорируется.
- Даже если перед значением года, месяца или дня имеется «0», данные все равно могут быть преобразованы.

Преобразование невозможно в указанных ниже случаях.

- Структура данных отличается от указанной выше.
- Такой даты не существует.

Например, значение строкового типа «2000-1-01» будет преобразовано в значение члена структуры D#2000-01-01.

● Дата и время

Ниже показана структура, которую должны иметь данные для преобразования в значение даты и времени.

Параметр	Описание
Год, месяц и день	Это значения года, месяца и дня, которые выражают дату.
Час	Диапазон допустимых значений: от 0 до 23.
Минуты	Диапазон допустимых значений: от 0 до 59.
Секунды	Диапазон допустимых значений: от 0 до 59,999999999. Если значение является целым числом, десятичная точка не требуется.
Дефисы и двоеточия	Эти элементы обязательны.

Примечание1. Знаки пробела (« ») перед значением года, месяца, дня, часа, минут или секунд игнорируются.

Примечание2. Если какие-либо символы в значении года, месяца, дня, часа, минут или секунд разделены одним символом подчеркивания («_»), этот символ подчеркивания игнорируется.

Примечание3. Даже если перед значением года, месяца, дня, часа, минут или секунд имеется «0», данные все равно могут быть преобразованы.

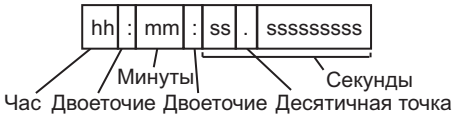
Преобразование невозможно в указанных ниже случаях.

- Структура данных отличается от указанной выше.
- Такой даты не существует.

Например, значение строкового типа «2000-01-23-4:56:07.89» будет преобразовано в значение члена структуры DT#2000-01-23-04:56:07.89.

● Время суток

Ниже показана структура, которую должны иметь данные для преобразования в значение времени суток.



Элемент	Описание
Час	Диапазон допустимых значений: от 0 до 23.
Минуты	Диапазон допустимых значений: от 0 до 59.
Секунды	Диапазон допустимых значений: от 0 до 59,999999999. Если значение является целым числом, десятичная точка («.») не требуется.
Двоеточия	Эти элементы обязательны.

Примечание1. Знаки пробела (« ») перед значением часа, минут или секунд игнорируются.

Примечание2. Если какие-либо символы в значении часа, минут или секунд разделены одним символом подчеркивания («_»), этот символ подчеркивания игнорируется.

Примечание3. Даже если перед значением часа, минут или секунд имеется «0», данные все равно могут быть преобразованы.

Преобразование невозможно в указанных ниже случаях.

- Структура данных отличается от указанной выше.
- Данные содержат несколько идущих подряд точек («.») или знаков подчеркивания («_»).

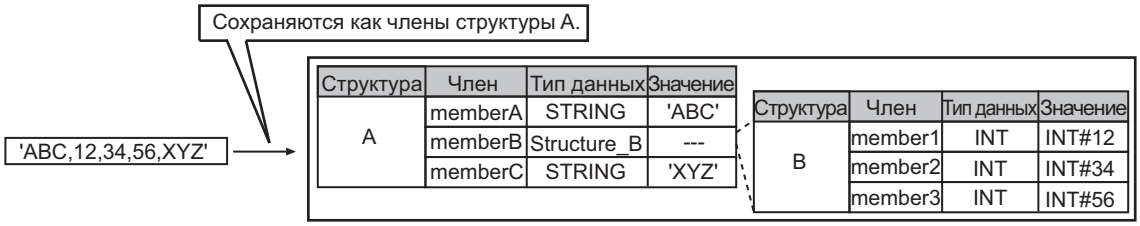
Например, значение строкового типа «12:23:34.567» будет преобразовано в значение члена структуры TOD#12:23:34.567.

● Текстовые строки

Значение члена структуры будет состоять из данных преобразуемой текстовой строки и добавленного в конце пустого символа (NULL). Но если длина текстовой строки превышает размер члена структуры, преобразование невозможно.
Например, значение строкового типа «ABC» без пустого символа (NULL) в конце будет преобразовано в значение члена структуры «ABC» с пустым символом (NULL) в конце.

● Структуры

Строковые данные (STRING) преобразуются в соответствии с правилами преобразования, применяемыми к типам данных членов структуры. Данные преобразуются по порядку, начиная с начала строки, и сохраняются как значения членов структуры до нижележащих (вложенных) уровней, данные на которых не являются структурами.
Например, если членом структуры A является структура B, преобразование выполняется показанным ниже образом.



● Перечисления

Строковые данные (STRING), выражающие значение переменной типа DINT, преобразуются в перечислитель перечисления.

При преобразовании в значение типа DINT применяются те же правила, что и для целых чисел. Значение типа DINT принимается за значение перечисления, и это значение преобразуется в соответствующий перечислитель.
Однако преобразование невозможно, если строковые данные не выражают значение типа DINT. В качестве примера рассмотрим перечисление *Color* с тремя перечислителями: *red*, *yellow* и *green*. Эти перечислители представляют следующие числовые значения: *red* = 1, *yellow* = 2, *green* = 3. Поэтому, например, строке «3» соответствует значение члена структуры *green*.

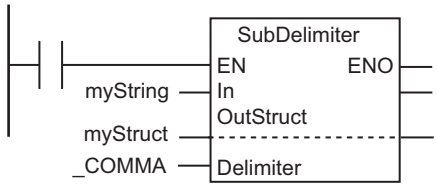
● Массивы

Каждый элемент данных, отделенный от других элементов, преобразуется в соответствующее значение элемента массива. Используются правила преобразования, применяемые для типа данных массива. Преобразование возможно, только если члены структуры являются одномерными массивами.
Допустим, к примеру, что некоторый член структуры является массивом myString[0..3] типа BYTE. При преобразовании разделенной запятыми текстовой строки «AA,BB,CC,DD» в элементы массива элементу myString[0] будет присвоено значение BYTE#16#AA, элементу myString[1] — значение BYTE#16#BB, элементу myString[2] — значение BYTE#16#CC, а элементу myString[3] — значение BYTE#16#DD.

Пример записи

Приведенный ниже пример демонстрирует, как разделенные запятыми данные в *myString* преобразуются и присваиваются членам структуры *myStruct*.

LD



ST

```
SubDelimiter(myString,myStruct,_COMMA);
```

myString 'XYZ,1234,aB,3.14'

Структура	Член	Тип данных	Значение
myStruct	member1	STRING	''
	member2	INT	INT#0
	member3	WORD	WORD#16#0000
	member4	REAL	REAL#0

Данные из myString сохраняются как значения членов структуры myStruct.

Структура	Член	Тип данных	Значение
myStruct	member1	STRING	'XYZ'
	member2	INT	INT#1234
	member3	WORD	WORD#16#00AB
	member4	REAL	REAL#3.14e+0

Дополнительная информация

- Данную команду можно использовать в сочетании с командой *FileGets* на стр. 2-1577 для простого чтения значений из указанных файлов CSV на карте памяти SD. Пример применения команды см. в разделе *Пример программы* на стр. 2-670.
- С помощью этой команды текстовую строку, полученную ранее из данных структурного типа с использованием команды *AddDelimiter* на стр. 2-651, можно снова преобразовать в данные структурного типа.

Меры предосторожности для обеспечения надлежащей эксплуатации

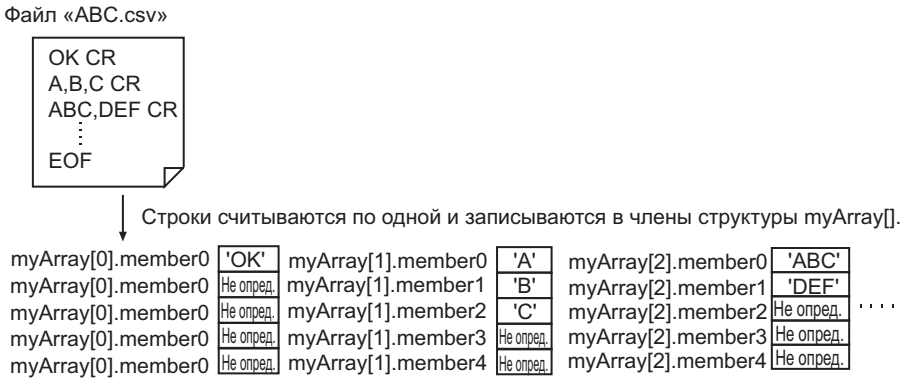
- При наличии двух и более идущих подряд разделителей в строке *In* соответствующие данные считаются несуществующими. Если данных не существует, значение члена структуры *OutStruct* будет неопределенным.
- Не используйте разделители для каких-либо других целей в переменной *In*. Даже если знак разделителя используется не в качестве разделителя, команда все равно воспринимает его как разделитель.
- Если в структуре *OutStruct* имеется член строкового типа (STRING), не добавляйте пустой символ (NULL) в конец соответствующего значения в строке *In*. Если где-либо в строке *In* (за исключением конца строки) имеется пустой символ (NULL), будут преобразованы только строковые данные до этого пустого символа.
- Если в структуре *OutStruct* имеется перечисление, убедитесь, что соответствующие данные в строке *In* выражают значение, которое определено как перечислитель. Даже если значение переменной перечислимого типа не определено как перечислитель, ошибки не произойдет.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* перейдет в состояние ЛОЖЬ, а значения в *OutStruct* будут неопределенными.
  a) Преобразование в тип данных члена структуры *OutStruct* выполнить невозможно.
  b) Результат преобразования выходит за диапазон допустимых значений соответствующего члена структуры *OutStruct*.
  c) Член структуры *OutStruct* является массивом с несколькими измерениями.
  d) Член структуры *OutStruct* является объединением.
  e) Размер *OutStruct* превышает 8192 байт.

Пример программы

В данном примере обрабатывается файл с именем «ABC.csv», который содержит несколько строк, разделяемых символами возврата каретки (т. е. кодами CR). Каждая строка файла представляет собой текстовую строку, элементы данных в которой отделяются друг от друга запятыми.

Программа считывает по одной строке файла за раз и сохраняет содержащиеся в строке данные, разделенные запятыми, в соответствующие элементы переменной-массива myArray[]. Каждый элемент этого массива является структурой типа *myStruct*. Данные сохраняются в каждый элемент как значения членов структуры, начиная с начала структуры. Структура *myStruct* содержит пять членов, которые являются переменными типа STRING.

Обработка завершается, когда оказываются прочитаны все данные до конца файла (т. е. когда оказывается прочитан код EOF).



Обработка выполняется в следующем порядке:

1. Используется команда FileOpen для открытия файла «ABC.csv».
2. Используется команда FileGets для чтения одной строки из файла.
3. Используется команда SubDelimiter для сохранения разделенных запятыми текстовых строк как значений членов структуры в соответствующий элемент массива myArray[].
4. Шаги 2 и 3 повторяются до достижения кода EOF (конца файла).
5. Используется команда FileClose для закрытия файла.

Определение типов данных

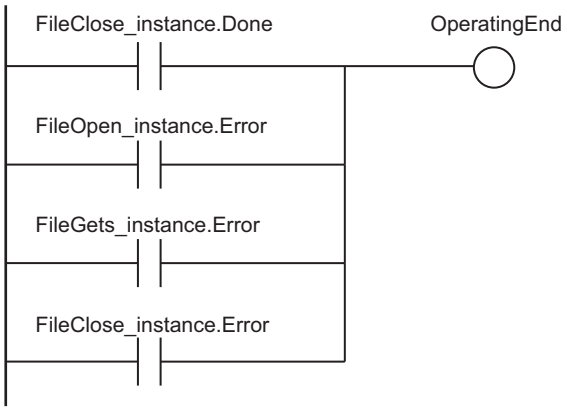
Имя	Тип данных	Комментарий
myStruct	STRUCT	Структура
member0	STRING	Член перечисления
member1	STRING	Член перечисления
member2	STRING	Член перечисления
member3	STRING	Член перечисления
member4	STRING	Член перечисления

Программа на языке LD

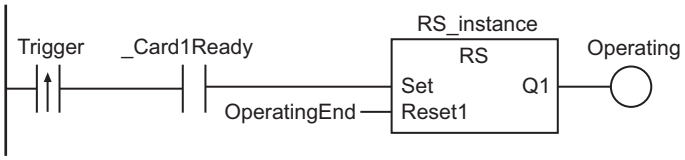
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	Ложь	Обработка завершена
	Trigger	BOOL	Ложь	Условие выполнения
	Operating	BOOL	Ложь	Обработка
	Index	INT	0	Индекс элемента myArray[]
	Fid	DWORD	16#0	ID файла
	myArray	ARRAY[0..999] OF myStruct	[1000((member0:='',member1:='',member2:='',member3:='',member4:=''))]	Целочисленные значения
	RS_instance	RS		
	FileOpen_instance	FileOpen		
	FileGets_instance	FileGets		
	FileClose_instance	FileClose		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

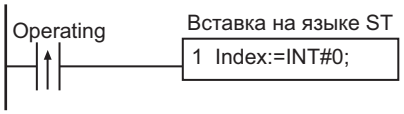
Определение, завершено ли выполнение команды.



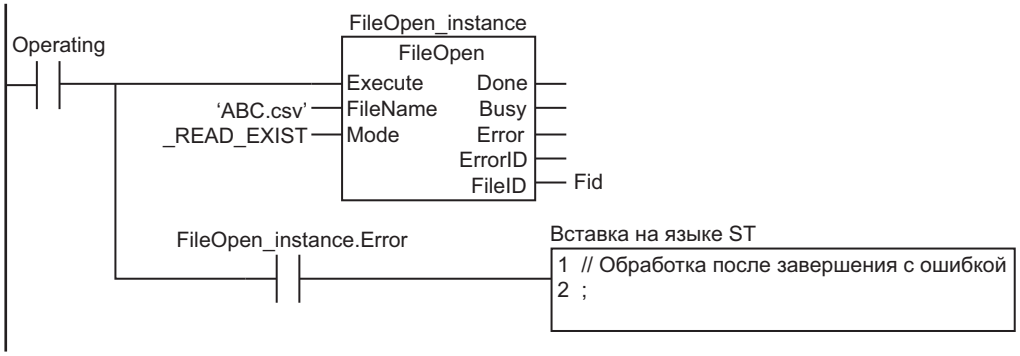
Прием условия выполнения.



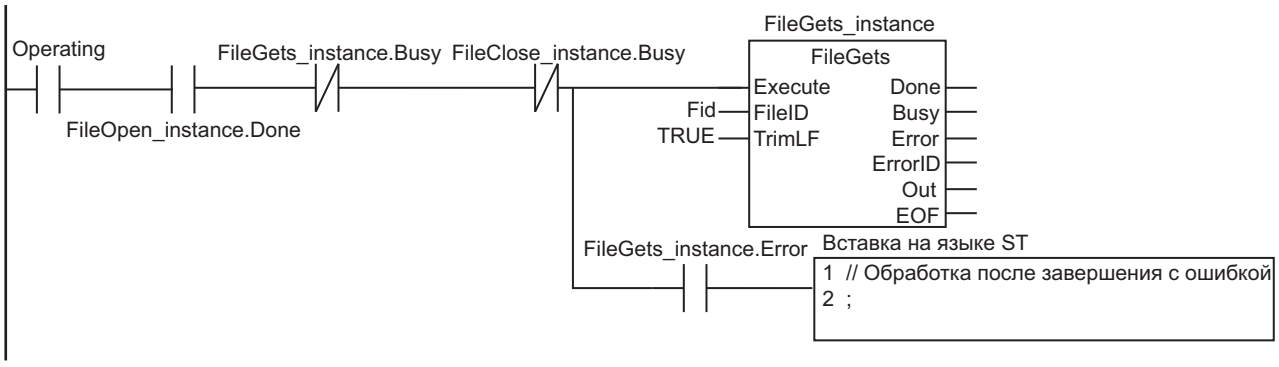
Инициализация индекса элемента ***InDat[]***.



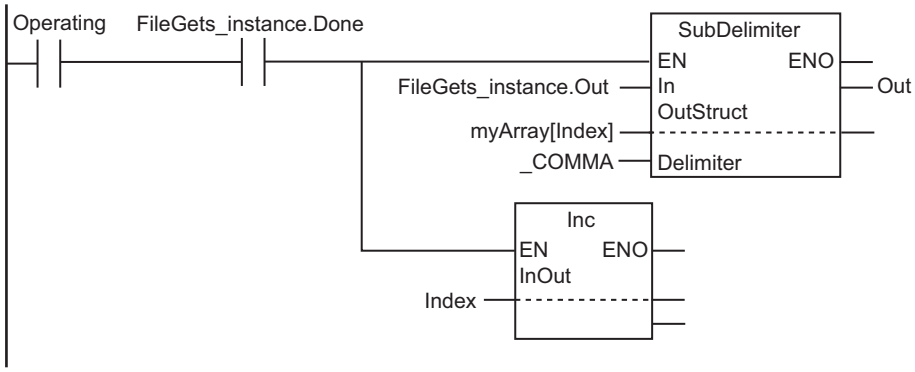
Выполнение команды FileOpen.



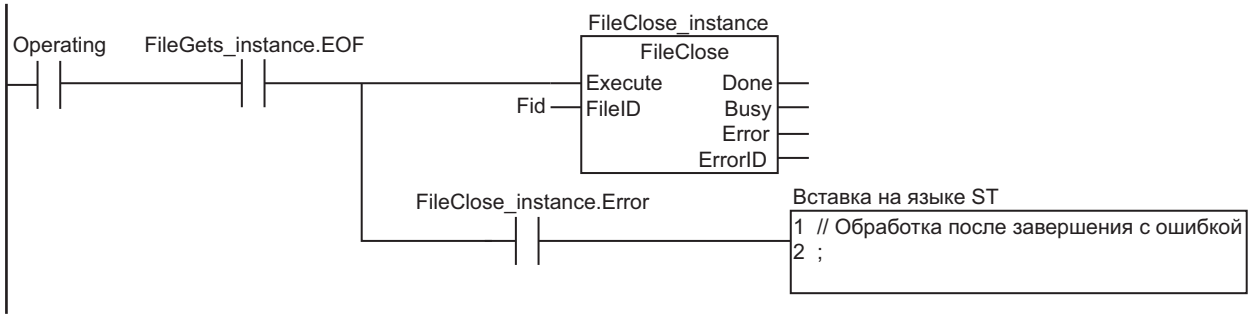
Выполнение команды FileGets.



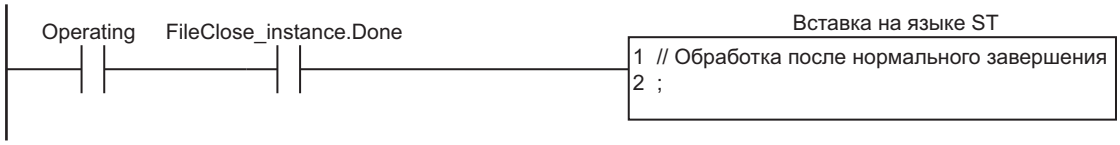
Выполнение команды SubDelimiter.



Выполнение команды FileClose при обнаружении EOF.



Обработка после нормального завершения



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	Ложь	Условие выполнения
	LastTrigger	BOOL	Ложь	Значение *Trigger* в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	Ложь	Обработка началась
	Operating	BOOL	Ложь	Обработка
	myArray	ARRAY[0..999] OF myStruct	[1000((member0:='',member1:='',member2:='',member3:='',member4:=''))]	Целочисленные значения
	Stage	INT	0	Смена этапа
	Index	INT	0	Индекс элемента myArray[]
	Fid	DWORD	16#0	ID файла
	FileOpen_instance	FileOpen		

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	FileGets_instance	FileGets		
	FileClose_instance	FileClose		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
  OperatingStart:=TRUE;
  Operating :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация экземпляра.
IF (OperatingStart=TRUE) THEN
  FileOpen_instance(Execute:=FALSE);
  FileGets_instance(Execute:=FALSE);
  FileClose_instance(Execute:=FALSE);
  Stage :=INT#1;
  Index :=INT#0;
  OperatingStart:=FALSE;
END_IF;

// Выполнение команды.
IF (Operating=TRUE) THEN
  CASE Stage OF
  1 : // Открытие файла.
    FileOpen_instance(
      Execute :=TRUE,
      FileName:='ABC.csv', // Имя файла
      Mode :=_READ_EXIST, // Чтение файла
      FileID =>Fid); // File ID

    IF (FileOpen_instance.Done=TRUE) THEN
      Stage:=INT#2; // Нормальное завершение
    END_IF;

    IF (FileOpen_instance.Error=TRUE) THEN
      Stage:=INT#99; // Завершение с ошибкой
    END_IF;

  2 : // Чтение текстовой строки.
    FileGets_instance(
      Execute:=TRUE,
      FileID :=Fid,
      TrimLF :=TRUE);
```

```
    IF (FileGets_instance.Done=TRUE) THEN
      // Сохранение прочитанных текстовых строк как значений членов структуры в myA
rray[].
      SubDelimiter(FileGets_instance.Out,myArray[Index],_COMMA);
      Index:=Index+INT#1;

      // Достигнут конец файла.
      IF (FileGets_instance.EOF=TRUE) THEN
        Stage:=INT#3; // Нормальное завершение
      ELSE
        FileGets_instance(Execute:=FALSE);
      END_IF;
    END_IF;

    IF (FileGets_instance.Error=TRUE) THEN
      Stage:=INT#99; // Завершение с ошибкой
    END_IF;

  3 : // Закрытие файла.
    FileClose_instance(
      Execute:=TRUE,
      FileID :=Fid); // Идентификатор файла

    IF (FileClose_instance.Done=TRUE) THEN
      Operating:=FALSE; // Нормальное завершение
    END_IF;

    IF (FileClose_instance.Error=TRUE) THEN
      Stage:=INT#99; // Завершение с ошибкой
    END_IF;

  99 : // Обработка после завершения с ошибкой
    Operating:=FALSE;
  END_CASE;
END_IF;
```

Команды для обработки времени и времени суток

Команда	Имя	Стр.
ADD_TIME	Сложение времени	стр. 2-679
ADD_TOD_TIME	Сложение времени с временем суток	стр. 2-681
ADD_DT_TIME	Сложение времени с датой и временем	стр. 2-683
SUB_TIME	Вычитание времени	стр. 2-685
SUB_TOD_TIME	Вычитание времени из времени суток	стр. 2-687
SUB_TOD_TOD	Вычитание времени суток	стр. 2-689
SUB_DATE_DATE	Вычитание даты	стр. 2-691
SUB_DT_DT	Вычитание даты и времени	стр. 2-693
SUB_DT_TIME	Вычитание времени из даты и времени	стр. 2-695
MULTIME	Умножение времени	стр. 2-697
DIVTIME	Деление времени	стр. 2-699
CONCAT_DATE_TOD	Объединение даты и времени суток	стр. 2-701
DT_TO_TOD	Извлечение времени суток из даты и времени	стр. 2-703
DT_TO_DATE	Извлечение даты из даты и времени	стр. 2-705
SetTime	Установка времени	стр. 2-707
GetTime	Получить время суток	стр. 2-709
DtToSec	Преобразование даты и времени в секунды	стр. 2-711
DateToSec	Преобразование даты в секунды	стр. 2-713
TodToSec	Преобразование времени суток в секунды	стр. 2-715
SecToDt	Преобразование секунд в дату и время	стр. 2-717
SecToDate	Преобразование секунд в дату	стр. 2-719
SecToTod	Преобразование секунд в время суток	стр. 2-721
TimeToNanoSec	Преобразование времени в наносекунды	стр. 2-723

Команда	Имя	Стр.
TimeToSec	Преобразование времени в секунды	стр. 2-725
NanoSecToTime	Преобразование наносекунд во время	стр. 2-727
SecToTime	Преобразование секунд во время	стр. 2-729
ChkLeapYear	Проверка на високосный год	стр. 2-731
GetDaysOfMonth	Получить количество дней в месяце	стр. 2-733
DaysToMonth	Преобразование дней в месяц	стр. 2-736
GetDayOfWeek	Получить день недели	стр. 2-738
GetWeekOfYear	Получить номер недели	стр. 2-740
DtToDateStruct	Разбивка даты и времени	стр. 2-742
DateStructToDt	Объединение времени	стр. 2-745
TruncTime	Усечение времени	стр. 2-748
TruncDt	Усечение даты и времени	стр. 2-753
TruncTod	Усечение времени суток	стр. 2-757

ADD_TIME

Команда ADD_TIME складывает два значения времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ADD_TIME	Сложение времени	FUN	(@)ADD_TIME EN ENO In1 Out In2	Out:=ADD_TIME(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Слагаемое-время 1	Вход	Слагаемое-время 1	Зависит от типа данных.	нс	T#0s
In2	Слагаемое-время 2		Слагаемое-время 2			
Out	Суммарное время	Выход	Суммарное время	Зависит от типа данных.	нс	---

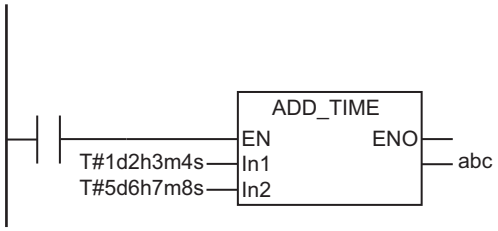
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																OK				
In2																OK				
Out																OK				

Функция

Команда ADD_TIME складывает два значения времени: *In1* и *In2*. Результат сложения в переменной *Out* также является значением времени.

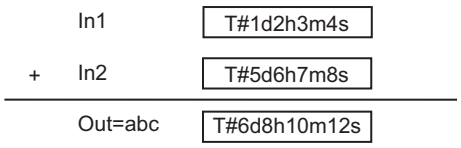
Ниже приведен пример, в котором *In1* = T#1d2h3m4s, а *In2* = T#5d6h7m8s.

LD



ST

```
abc:=ADD_TIME(T#1d2h3m4s, T#5d6h7m8s);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

Даже если результат сложения выйдет за диапазон допустимых значений *Out*, ошибки не произойдет. В примерах ниже показано, как в этом случае выполняется сложение.

- T#106751d_23h_47m_16s_854.775807ms + T#0.000001ms
  → T#-106751d_23h_47m_16s_854.775808ms
- T#-106751d_23h_47m_16s_854.775808ms + T#-0.000001ms
  → T#106751d_23h_47m_16s_854.775807ms

ADD_TOD_TIME

Команда ADD_TOD_TIME складывает значение времени со значением времени суток.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ADD_TOD_TIME	Сложение времени с временем суток	FUN	(@)ADD_TOD_TIME EN ENO In1 Out In2	Out:=ADD_TOD_TIME(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Слагаемое-время суток	Вход	Слагаемое-время суток	Зависит от типа данных.	Час, минуты, секунды	TOD#0:0:0
In2	Слагаемое-время		Слагаемое-время		нс	T#0s
Out	Результирующее время суток	Выход	Результирующее время суток	Зависит от типа данных.	Час, минуты, секунды	---

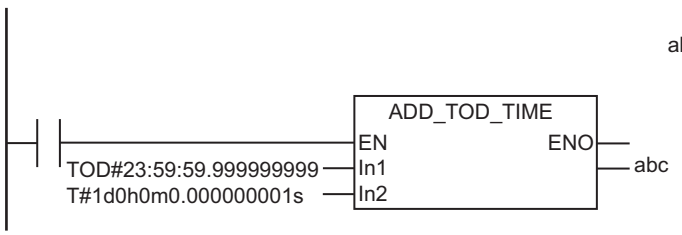
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																		ОК		
In2																ОК				
Out																		ОК		

Функция

Команда ADD_TOD_TIME складывает значение времени *In2* со значением времени суток *In1*. Результат сложения в переменной *Out* также является значением времени суток.

Ниже приведен пример, в котором *In1* = TOD#23:59:59.999999999, а *In2* = T#1d0h0m0.000000001s.

LD



ST

```
abc:=ADD_TOD_TIME(TOD#23:59:59.999999999,
                  T#1d0h0m0.000000001s);
```

	In1	TOD#23:59:59.999999999
+	In2	T#1d0h0m0.000000001s
	Out=abc	TOD#0:0:0.000000000

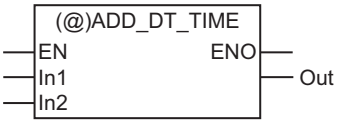
Меры предосторожности для обеспечения надлежащей эксплуатации

Даже если результат сложения выйдет за диапазон допустимых значений *Out*, ошибки не произойдет. В примерах ниже показано, как в этом случае выполняется сложение.

- TOD#23:59:59.999999999 + T#0.000001ms → TOD#0:0:0.000000000
- TOD#0:0:0.000000000 + T#-0.000001ms → TOD#23:59:59.999999999

ADD_DT_TIME

Команда ADD_DT_TIME складывает значение времени со значением даты и времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ADD_DT_TIME	Сложение времени с датой и временем	FUN		Out:=ADD_DT_TIME(In1, In2);

Переменные

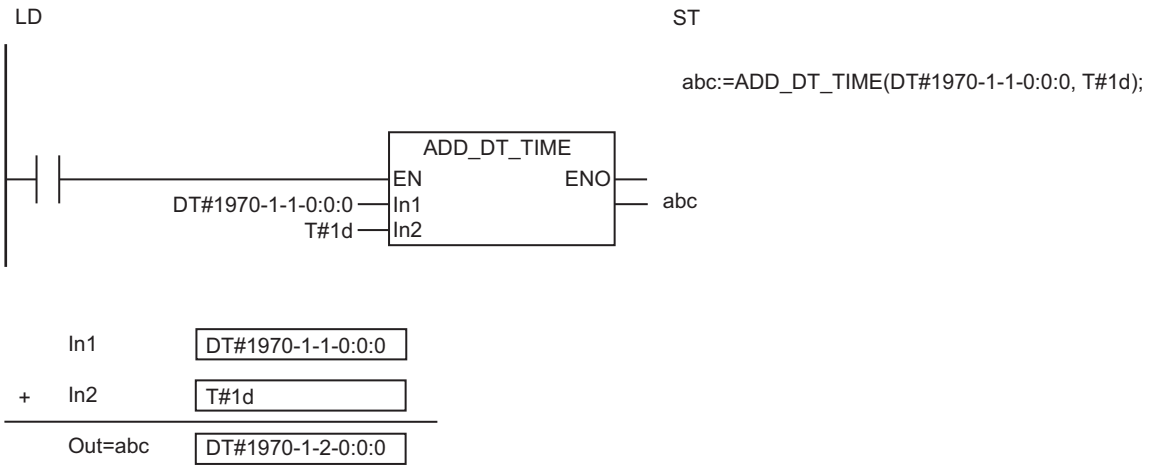
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Слагаемое-дата и время	Вход	Слагаемое-дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	DT#1970-1-1-0:0:0
In2	Слагаемое-время		Слагаемое-время		нс	T#0s
Out	Дата и время в результате сложения	Выход	Дата и время в результате сложения	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																			ОК	
In2																ОК				
Out																			ОК	

Функция

Команда ADD_DT_TIME складывает значение времени *In2* со значением даты и времени *In1*. Результат сложения в переменной *Out* является значением даты и времени. Команда также принимает в расчет високосные годы.

Ниже приведен пример, в котором *In1* = DT#1970-1-1-0:0:0, а *In2* = T#1d.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

Меры предосторожности для обеспечения надлежащей эксплуатации

Даже если результат сложения выйдет за диапазон допустимых значений *Out*, ошибки не произойдет. В примерах ниже показано, как в этом случае выполняется сложение.

- DT#2554-7-21-23:34:33.709551615 + T#0.000001ms → DT#1970-1-1-0:0:0
- DT#1970-1-1-0:0:0 + T#-0.000001ms → DT#2554-7-21-23:34:33.709551615

SUB_TIME

Команда SUB_TIME вычитает значение времени из другого значения времени.

Команда	Имя	FB/FUN	Графическое представление	Выражение языка ST
SUB_TIME	Вычитание времени	FUN	(@)SUB_TIME EN ENO In1 Out In2	Out:=SUB_TIME(In1, In2);

Переменные

	Значение	Вход/выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Исходное время	Вход	Исходное время	Зависит от типа данных.	нс	T#0s
In2	Вычитаемое время		Вычитаемое время			
Out	Результирующее время	Выход	Результирующее время	Зависит от типа данных.	нс	---

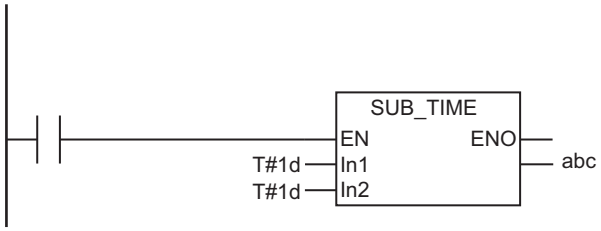
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																OK				
In2																OK				
Out																OK				

Функция

Команда SUB_TIME вычитает значение времени *In2* из значения времени *In1*. Результат вычитания в переменной *Out* также является значением времени.

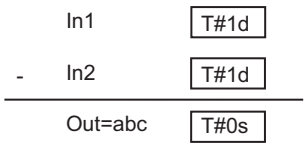
Ниже приведен пример, в котором *In1* и *In2* равны T#1d.

LD



ST

```
abc:=SUB_TIME(T#1d, T#1d);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

Даже если результат вычитания выйдет за диапазон допустимых значений *Out*, ошибки не произойдет. В примерах ниже показано, как в этом случае выполняется вычитание.

- T#106751d_23h_47m_16s_854.775807ms - T#-0.000001ms
  → T#-106751d_23h_47m_16s_854.775808ms
- T#-106751d_23h_47m_16s_854.775808ms - T#0.000001ms
  → T#106751d_23h_47m_16s_854.775807ms

SUB_TOD_TIME

Команда SUB_TOD_TIME вычитает значение времени из значения времени суток.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SUB_TOD_TIME	Вычитание времени из времени суток	FUN	(@)SUB_TOD_TIME EN ENO In1 Out In2	Out:=SUB_TOD_TIME(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Время суток	Вход	Время суток	Зависит от типа данных.	Час, минуты, секунды	TOD#0:0:0
In2	Вычитаемое время		Вычитаемое время		нс	T#0s
Out	Результирующее время суток	Выход	Результирующее время суток	Зависит от типа данных.	Час, минуты, секунды	---

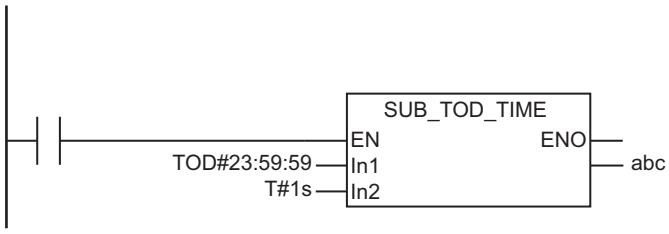
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																		ОК		
In2																ОК				
Out																		ОК		

Функция

Команда SUB_TOD_TIME вычитает значение времени *In2* из значения времени суток *In1*. Результат вычитания в переменной *Out* является значением времени суток.

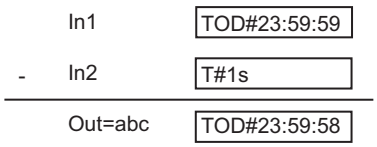
Ниже приведен пример, в котором *In1* = TOD#23:59:59, а *In2* = T#1s.

LD



ST

```
abc:=SUB_TOD_TIME(TOD#23:59:59, T#1s);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

Даже если результат вычитания выйдет за диапазон допустимых значений *Out*, ошибки не произойдет. В примерах ниже показано, как в этом случае выполняется вычитание.

- TOD#23:59:59.999999999 - T#-0.000001ms → TOD#0:0:0
- TOD#0:0:0 - T#0.000001ms → TOD#23:59:59.999999999

SUB_TOD_TOD

Команда SUB_TOD_TOD вычитает значение времени суток из другого значения времени суток.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SUB_TOD_TOD	Вычитание времени суток	FUN	(@)SUB_TOD_TOD EN ENO In1 Out In2	Out:=SUB_TOD_TOD(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Время суток 1	Вход	Время суток 1	Зависит от типа данных.	Час, минуты, секунды	TOD#0:0:0
In2	Время суток 2		Время суток 2			
Out	Результирующее время	Выход	Результирующее время	Зависит от типа данных.	нс	---

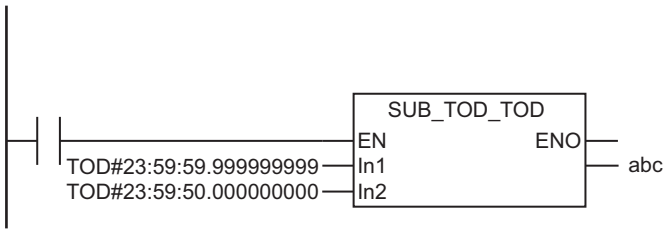
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																		OK		
In2																		OK		
Out																OK				

Функция

Команда SUB_TOD_TOD вычитает значение времени суток *In2* из значения времени суток *In1*. Результат вычитания в переменной *Out* является значением времени.

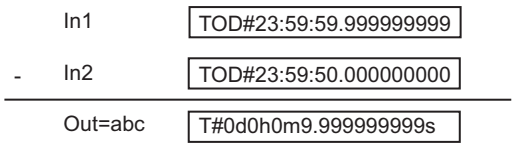
Ниже приведен пример, в котором *In1* = TOD#23:59:59.999999999, а *In2* = TOD#23:59:50.000000000.

LD



ST

```
abc:=SUB_TOD_TOD(TOD#23:59:59.999999999,
                 TOD#23:59:50.000000000);
```


In1
TOD#23:59:59.999999999
-
In2
TOD#23:59:50.000000000
Out=abc
T#0d0h0m9.999999999s

SUB_DATE_DATE

Команда SUB_DATE_DATE вычитает значение даты из другого значения даты.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SUB_DATE_DATE	Вычитание даты	FUN	(@)SUB_DATE_DATE EN ENO In1 Out In2	Out:=SUB_DATE_DATE(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Дата 1	Вход	Дата 1	Зависит от типа данных.	Год, месяц, день	DT#1970-1-1
In2	Дата 2		Дата 2			
Out	Результирующее время	Выход	Результирующее время	Зависит от типа данных.	нс	---

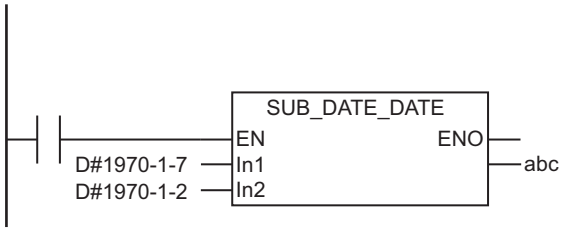
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																	OK			
In2																	OK			
Out																OK				

Функция

Команда SUB_DATE_DATE вычитает дату *In2* из даты *In1*. Результат вычитания в переменной *Out* является значением времени.

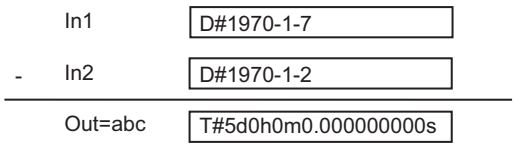
Ниже приведен пример, в котором *In1* = D#1970-1-7, а *In2* = D#1970-1-2.

LD



ST

```
abc:=SUB_DATE_DATE(D#1970-1-7, D#1970-1-2);
```


In1
D#1970-1-7
-
In2
D#1970-1-2
Out=abc
T#5d0h0m0.000000000s

SUB_DT_DT

Команда SUB_DT_DT вычитает значение даты и времени из другого значения даты и времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SUB_DT_DT	Вычитание даты и времени	FUN	(@)SUB_DT_DT EN ENO In1 Out In2	Out:=SUB_DT_DT(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Дата и время 1	Вход	Дата и время 1	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	DT#1970-1-1-0:0:0
In2	Дата и время 2		Дата и время 2			
Out	Результирующее время	Выход	Результирующее время	Зависит от типа данных.	нс	---

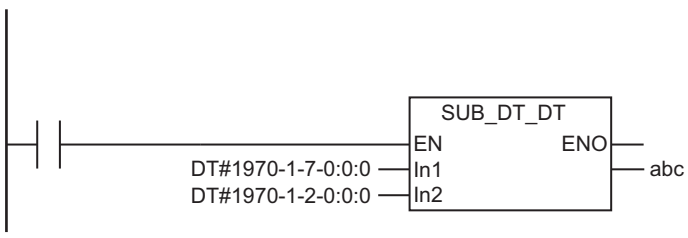
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																			OK	
In2																			OK	
Out																OK				

Функция

Команда SUB_DT_DT вычитает значение даты и времени *In2* из значения даты и времени *In1*. Результат вычитания в переменной *Out* является значением времени.

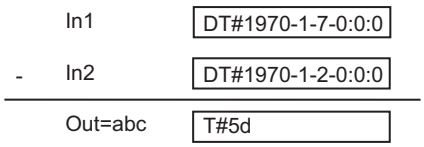
Ниже приведен пример, в котором *In1* = DT#1970-1-7-0:0:0, а *In2* = DT#1970-1-2-0:0:0.

LD



ST

```
abc:=SUB_DT_DT(DT#1970-1-7-0:0:0,
               DT#1970-1-2-0:0:0);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

Меры предосторожности для обеспечения надлежащей эксплуатации

Если результат обработки выйдет за диапазон допустимых значений *Out*, *Out* будет содержать недопустимое значение.

SUB_DT_TIME

Команда SUB_DT_TIME вычитает значение времени из значения даты и времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SUB_DT_TIME	Вычитание времени из даты и времени	FUN	(@)SUB_DT_TIME EN ENO In1 Out In2	Out:=SUB_DT_TIME(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Дата и время	Вход	Дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	DT#1970-1-1-0:0:0
In2	Вычитаемое время		Вычитаемое время		нс	T#0s
Out	Результирующие дата и время	Выход	Результирующие дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	---

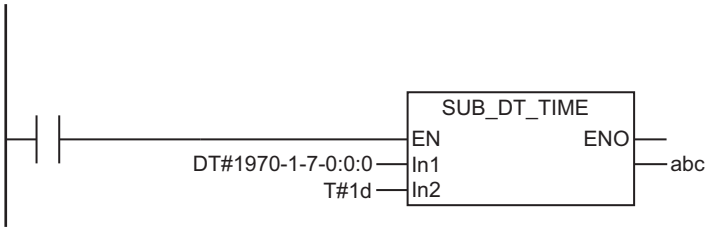
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																			OK	
In2																OK				
Out																			OK	

Функция

Команда SUB_DT_TIME вычитает значение времени *In2* из значения даты и времени *In1*. Результат вычитания в переменной *Out* является значением даты и времени. Команда также принимает в расчет високосные годы.

Ниже приведен пример, в котором *In1* = DT#1970-1-1-0:0:0, а *In2* = T#1d.

LD



ST

```
abc:=SUB_DT_TIME(DT#1970-1-7-0:0:0, T#1d);
```

	In1	DT#1970-1-7-0:0:0
-	In2	T#1d
	Out=abc	DT#1970-1-6-0:0:0

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

Меры предосторожности для обеспечения надлежащей эксплуатации

Даже если результат вычитания выйдет за диапазон допустимых значений *Out*, ошибки не произойдет. В примерах ниже показано, как в этом случае выполняется вычитание.

- DT#2554-7-21-23:34:33.709551615 - T#-0.000001ms → DT#1970-1-1-0:0:0
- DT#1970-1-1-0:0:0 - T#0.000001ms → DT#2554-7-21-23:34:33.709551615

MULTIME

Команда MULTIME умножает значение времени на заданное число.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
MULTIME	Умножение времени	FUN	(@)MULTIME EN ENO In1 Out In2	Out:=MULTIME(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Исходное время	Вход	Исходное время	Зависит от типа данных.	нс	T#0s
In2	Множитель		Множитель		---	*1
Out	Результирующее время	Выход	Результирующее время	Зависит от типа данных.	нс	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

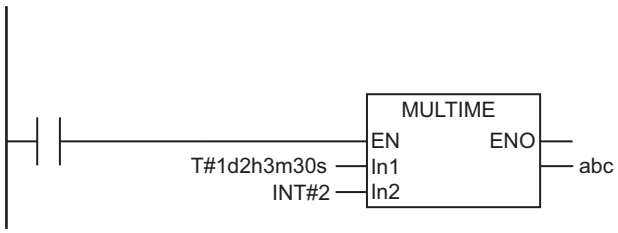
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																OK				
In2						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
Out																OK				

Функция

Команда MULTIME умножает значение времени *In1* на множитель *In2*. Результат умножения в переменной *Out* также является значением времени.

Ниже приведен пример, в котором *In1* = T#1d2h3m30s, а *In2* = INT#2.

LD



ST

```
abc:=MULTIME(T#1d2h3m30s, INT#2);
```

	In1	T#1d2h3m30s
×	In2	INT#2
	Out=abc	T#2d4h7m

Меры предосторожности для обеспечения надлежащей эксплуатации

- Если *In2* является вещественным числом, результат умножения округляется до девятого знака после запятой в значении секунд (т. е. до наносекунд).
- В таблице ниже приведены значения *Out* для случаев, когда *In2* равно 0 либо является положительной бесконечностью, отрицательной бесконечностью или нечисловым значением.

Значение *In2*	Значение *Out*	
	NX1P2	Прочее (кроме указанного слева)
0	T#0s	T#0s
+∞	T#-0d0h0m0s1e-6ms	T#-106751d23h47m16.854775808s
-∞	T#-0d0h0m0s1e-6ms	T#-106751d23h47m16.854775808s
Нечисловое значение	T#0s	T#-106751d23h47m16.854775808s

- Даже если результат умножения выйдет за диапазон допустимых значений *Out*, ошибки не произойдет. В примерах ниже показано, как в этом случае выполняется умножение.
  a) T#53375d_23h_53m_38s_427.387904ms * USINT#2
  → T#-106751d_23h_47m_16s_854.775808ms
  b) T#-53375d_23h_53m_38s_427.387905ms * USINT#2
  → T#106751d_23h_47m_16s_854.775806ms

Округление

В следующей таблице показано, как округляются значения.

Значение дробной части	Описание	Примеры
Меньше 0,5	Дробная часть отбрасывается.	1,49 → 1
0,5	Если разряд единиц содержит четное число, дробная часть отбрасывается. Если это нечетное число, то значение округляется в большую сторону.	1,50 → 2 2,50 → 2
Больше 0,5	Дробная часть отбрасывается, число округляется в большую сторону.	1,51 → 2

DIVTIME

Команда DIVTIME делит значение времени на заданное число.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DIVTIME	Деление времени	FUN	(@)DIVTIME EN ENO In1 Out In2	Out:=DIVTIME(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Исходное время	Вход	Исходное время	Зависит от типа данных.	нс	T#0s
In2	Число-делитель		Число-делитель		---	*1
Out	Результирующее время	Выход	Результирующее время	Зависит от типа данных.	нс	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

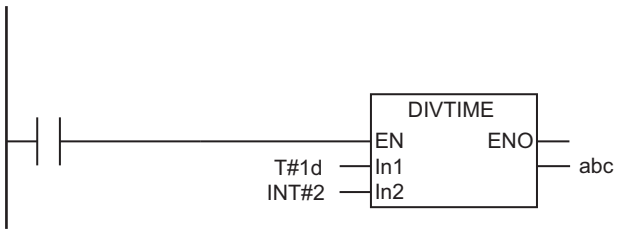
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																OK				
In2						OK	OK	OK	OK	OK	OK	OK	OK	OK	OK					
Out																OK				

Функция

Команда DIVTIME делит значение времени *In1* на число *In2*. Результат деления в переменной *Out* также является значением времени.

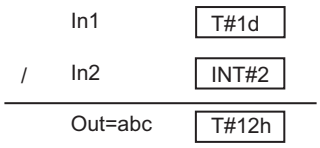
Ниже приведен пример, в котором *In1* = T#1d, а *In2* = INT#2.

LD



ST

```
abc:=DIVTIME(T#1d, INT#2);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

- В таблице ниже приведены значения *Out* для случаев, когда *In2* равно 0 либо является положительной бесконечностью, отрицательной бесконечностью или нечисловым значением.

Значение *In2*	Значение *Out*	
	NX1P2	Прочее (кроме указанного слева)
0	T#0d_0h_0m_0s_1e-006	T#-106751d23h47m16.854775808s
+∞	T#0s	T#0s
-∞	T#0s	T#0s
Нечисловое значение	T#0s	T#-106751d23h47m16.854775808s

- Если *In2* является вещественным числом, возможна ошибка до нескольких наносекунд.
- Если *In2* является вещественным числом, результат деления округляется до девятого знака после запятой в значении секунд (т. е. до наносекунд).
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) *In2* является целым числом со значением 0.

Округление

В следующей таблице показано, как округляются значения.

Значение дробной части	Описание	Примеры
Меньше 0,5	Дробная часть отбрасывается.	1,49 → 1
0,5	Если разряд единиц содержит четное число, дробная часть отбрасывается. Если это нечетное число, то значение округляется в большую сторону.	1,50 → 2 2,50 → 2
Больше 0,5	Дробная часть отбрасывается, число округляется в большую сторону.	1,51 → 2

CONCAT_DATE_TOD

Команда CONCAT_DATE_TOD объединяет значение даты со значением времени суток.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CONCAT_DATE_TOD	Объединение даты и времени суток	FUN	(@)CONCAT_DATE_TOD EN In1 In2 ENO Out	Out:=CONCAT_DATE_TOD(In1, In2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In1	Дата	Вход	Дата	Зависит от типа данных.	Год, месяц, день	DT#1970-1-1
In2	Время суток		Время суток		Час, минуты, секунды	TOD#0:0:0
Out	Объединенные значения даты и времени	Выход	Объединенные значения даты и времени	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	---

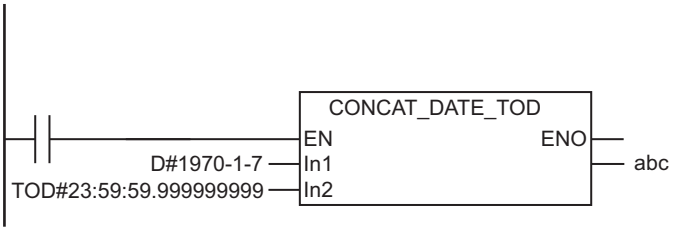
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																	ОК			
In2																		ОК		
Out																			ОК	

Функция

Команда CONCAT_DATE_TOD объединяет значение даты *In1* со значением времени суток *In2*. Результат объединения в переменной *Out* является значением даты и времени.

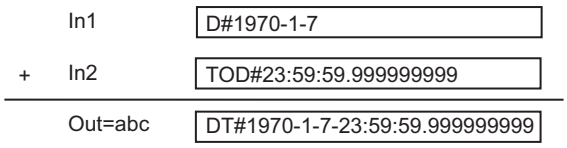
Ниже приведен пример, в котором *In1* = D#1970-1-7, а *In2* = TOD#23:59:59.999999999.

LD



ST

```
abc:=CONCAT_DATE_TOD(D#1970-1-7,
                     TOD#23:59:59.999999999);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

Меры предосторожности для обеспечения надлежащей эксплуатации

В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Полученное в результате объединения значение даты и времени выходит за диапазон допустимых значений переменной *Out*
  (например, если *In1* = D#2554-7-21, а значение *In2* больше TOD#23:34:33.709551615, диапазон допустимых значений *Out* оказывается превышен).

DT_TO_TOD

Команда DT_TO_TOD извлекает значение времени суток из значения даты и времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DT_TO_TOD	Извлечение времени суток из даты и времени	FUN	(@)DT_TO_TOD EN ENO In Out	Out:=DT_TO_TOD(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Дата и время	Вход	Дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	DT#1970-1-1-0:0:0
Out	Время суток	Выход	Время суток	Зависит от типа данных.	Час, минуты, секунды	---

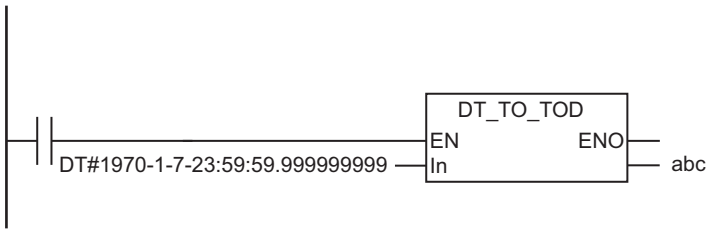
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																			OK	
Out																		OK		

Функция

Команда DT_TO_TOD извлекает значение времени суток из значения даты и времени *In*.

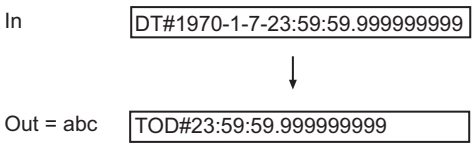
В приведенном ниже примере *In* = DT#1970-1-7-23:59:59.999999999.

LD



ST

```
abc:=DT_TO_TOD(
      DT#1970-1-7-23:59:59.999999999);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

DT_TO_DATE

Команда DT_TO_DATE извлекает значение даты из значения даты и времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DT_TO_DATE	Извлечение даты из даты и времени	FUN	(@)DT_TO_DATE EN ENO In Out	Out:=DT_TO_DATE(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Дата и время	Вход	Дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	DT#1970-1-1-0:0:0
Out	Дата	Выход	Дата	Зависит от типа данных.	Год, месяц, день	---

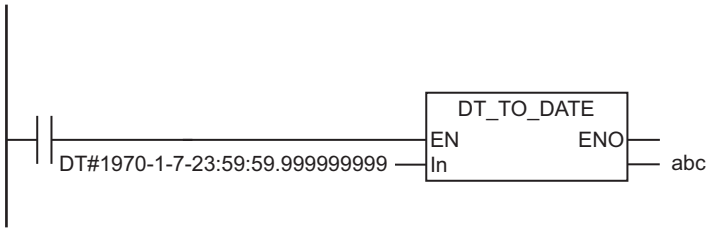
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																			OK	
Out																	OK			

Функция

Команда DT_TO_DATE извлекает значение даты из значения даты и времени *In*.

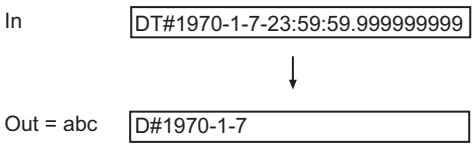
В приведенном ниже примере *In* = DT#1970-1-7-23:59:59.999999999.

LD



ST

```
abc:=DT_TO_DATE(
      DT#1970-1-7-23:59:59.999999999);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

SetTime

Команда SetTime устанавливает системное время.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SetTime	Установка времени	FUN	(@)SetTime EN ENO In Out	SetTime(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Данные времени	Вход	Текущее время для установки системного времени	*1	Год, месяц, день, час, минуты, секунды	DT#1970-1-1-0:0:0
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Ниже указаны диапазоны допустимых значений времени по Гринвичу (GMT).
Диапазон допустимых значений для модуля ЦПУ серии NX: DT#1970-01-01-00:00:00.000000000... DT#2069-12-31-23:59:59.999999999 (от 0:00:000000000, 1 января 1970 г. до 23:59:59.999999999 31 декабря 2069 г.).
Диапазон допустимых значений для модуля ЦПУ серии NJ: DT#1970-01-01-00:00:00.000000000...DT#2106-02-06-23:59:59.999999999 (от 0:00:000000000, 1 января 1970 г. до 23:59:59.999999999, 6 февраля 2106 г.).

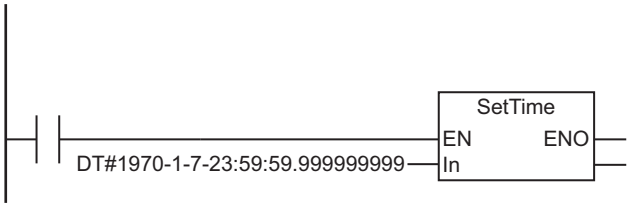
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																			OK	
Out	OK																			

Функция

Команда SetTime устанавливает системное время равным значению даты и времени в переменной *In*.

В показанном ниже примере программы *In* = DT#1970-1-7:23:59:59.999999999.

LD



ST

```
SetTime(DT#1970-1-7-23:59:59.999999999);
```

Команда SetTime устанавливает системное время суток равным значению ***In***.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

Дополнительная информация

Для установки системного времени также можно использовать указанные ниже способы.

- Sysmac Studio
- Функция NTP

Меры предосторожности для обеспечения надлежащей эксплуатации

- В переменной *In* следует указать время для установленного часового пояса (а не время по Гринвичу (GMT)).
- В переменной *In* нельзя задать время меньше 1970-1-1-0:0:0.000000000 GMT.
- Внутреннее время обновляется с некоторой задержкой. Если значение времени считывается сразу после выполнения этой команды, может быть прочитано старое значение времени.
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *In* находится за пределами допустимого диапазона.
  b) Значение *In* меньше 1970-1-1-0:0:0.000000000 по Гринвичу.

GetTime

Команда GetTime считывает текущее время.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetTime	Получить время суток	FUN	(@)GetTime EN ENO Out	Out:=GetTime();

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Текущее время	Выход	Текущее время	*1	Год, месяц, день, час, минуты, секунды	---

*1. Ниже указаны диапазоны допустимых значений времени по Гринвичу (GMT).
Диапазон допустимых значений для модуля ЦПУ серии NX: DT#1970-01-01-00:00:00.000000000... DT#2069-12-31-23:59:59.999999999 (от 0:00:000000000, 1 января 1970 г. до 23:59:59.999999999 31 декабря 2069 г.).
Диапазон допустимых значений для модуля ЦПУ серии NJ: DT#1970-01-01-00:00:00.000000000...DT#2106-02-06-23:59:59.999999999 (от 0:00:000000000, 1 января 1970 г. до 23:59:59.999999999, 6 февраля 2106 г.).

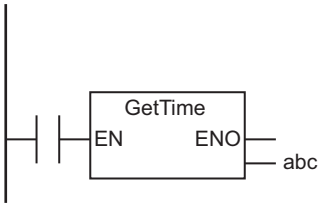
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out																			OK	

Функция

Команда GetTime считывает текущее время.
Считывается текущее время суток в установленном часовом поясе (а не время по Гринвичу (GMT)).

Пример программы представлен на рисунке ниже. Текущее время присваивается переменной *abc*.

LD

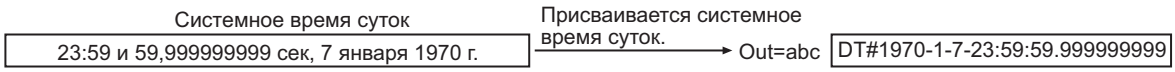


ST

```
abc:=GetTime();
```

Команда GetTime присваивает переменной ***abc*** текущее время.

Пример для 23:59 и 59,999999999 сек, 7 января 1970 г.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

Дополнительная информация

- Для преобразования текущего времени суток в системное время суток (количество секунд, начиная с 00:00:00, 1 января 1970 г.) используйте команду *DtToSec* на стр. 2-711.
- Для преобразования текущего времени суток в дату (год, месяц, день, минуты и секунды) используйте команду *DtToDateStruct* на стр. 2-742.
- Для считывания значения дня недели используйте команду *GetDayOfWeek* на стр. 2-738.

DtToSec

Команда DtToSec преобразует дату и время в количество секунд, начиная с 00:00:00, 1 января 1970 г.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DtToSec	Преобразование даты и времени в секунды	FUN	(@)DtToSec EN ENO In Out	Out:=DtToSec(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Дата и время	Вход	Дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	DT#1970-1-1-0:0:0
Out	Секунды	Выход	Количество секунд, начиная с 00:00:00, 1 января 1970 г.	0...18446744073	Секунды	---

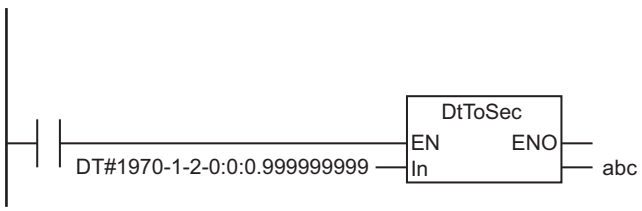
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																			OK	
Out													OK							

Функция

Команда DtToSec преобразует значение даты и времени *In* в значение количества секунд, начиная с 00:00:00, 1 января 1970 г. Результатом преобразования является количество секунд. Данные после значения секунд в полученном значении отбрасываются.

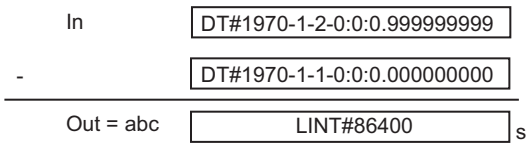
В приведенном ниже примере *In* = DT#1970-1-2-0:0:0.999999999.

LD



ST

```
abc:=DtToSec(DT#1970-1-2-0:0:0.999999999);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

Дополнительная информация

Для преобразования количества секунд, начиная с 00:00:00, 1 января 1970 г., в значение даты и времени используйте команду *SecToDt* на стр. 2-717.

DateToSec

Команда DateToSec преобразует дату в количество секунд, начиная с 00:00:00, 1 января 1970 г.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DateToSec	Преобразование даты в секунды	FUN	(@)DateToSec EN ENO In Out	Out:=DateToSec(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Дата	Вход	Дата	Зависит от типа данных.	Год, месяц, день	DT#1970-1-1
Out	Секунды	Выход	Количество секунд, начиная с 00:00:00, 1 января 1970 г.	0...18446659200	Секунды	---

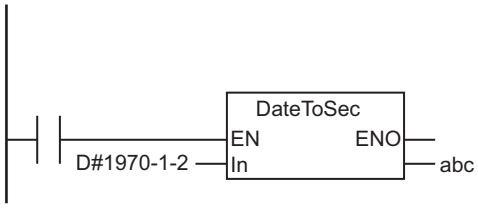
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																	OK			
Out													OK							

Функция

Команда DateToSec преобразует значение даты *In* (время принимается равным 00:00:00) в количество секунд, начиная с 00:00:00, 1 января 1970 г. Результатом преобразования является количество секунд.

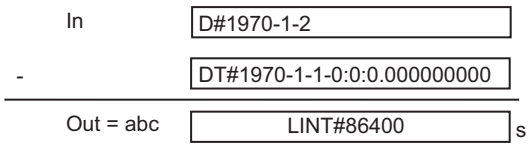
Ниже приведен пример для случая, когда *In* = D#1970-1-2.

LD



ST

```
abc:=DateToSec(D#1970-1-2);
```



Дополнительная информация

Для преобразования количества секунд, начиная с 00:00:00, 1 января 1970 г., в значение даты используйте команду *SecToDate* на стр. 2-719.

TodToSec

Команда TodToSec преобразует время суток в количество секунд, начиная с 00:00:00.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TodToSec	Преобразование времени суток в секунды	FUN	(@)TodToSec EN ENO In Out	Out:=TodToSec(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Время суток	Вход	Время суток	Зависит от типа данных.	Час, минуты, секунды	TOD#0:0:0
Out	Секунды	Выход	Количество секунд, начиная с 00:00:00	0...86399	Секунды	---

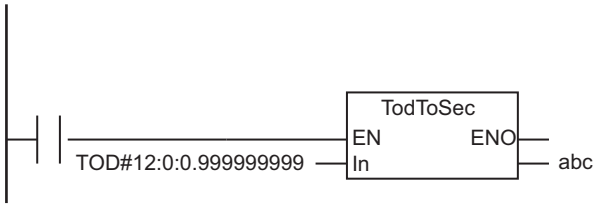
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																		OK		
Out													OK							

Функция

Команда TodToSec преобразует значение времени суток *In* в значение количества секунд, начиная с 00:00:00. Результатом преобразования является количество секунд. Данные после значения секунд в полученном значении отбрасываются.

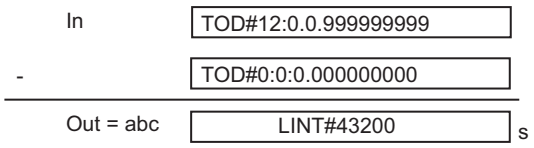
Ниже приведен пример для случая, когда *In* = TOD#12:0:0.999999999.

LD



ST

```
abc:=TodToSec(TOD#12:0:0.999999999);
```

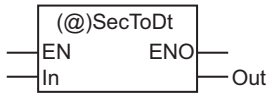


Дополнительная информация

Для преобразования количества секунд, начиная с 00:00:00, в значение времени суток используйте команду *SecToTod* на стр. 2-721.

SecToDt

Команда SecToDt преобразует количество секунд, начиная с 00:00:00, 1 января 1970 г., в значение даты и времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SecToDt	Преобразование секунд в дату и время	FUN		Out:=SecToDt(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Секунды	Вход	Количество секунд, начиная с 00:00:00, 1 января 1970 г.	0...18446744073	Секунды	0
Out	Дата и время	Выход	Дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	---

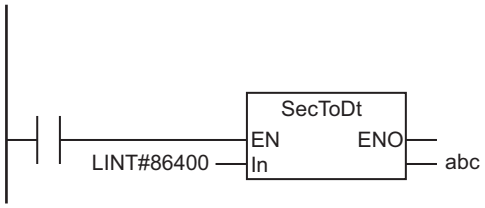
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In													OK							
Out																			OK	

Функция

Команда SecToDt преобразует значение количества секунд, начиная с 00:00:00, 1 января 1970 г., в значение даты и времени *In*.

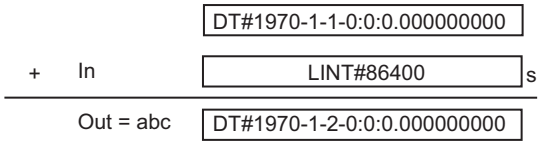
Ниже приведен пример для случая, когда *In* = LINT#86400.

LD



ST

```
abc:=SecToDt(LINT#86400);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

Дополнительная информация

Для преобразования текущего времени суток в количество секунд, начиная с 00:00:00, 1 января 1970 г., используйте команду *DtToSec* на стр. 2-711.

Меры предосторожности для обеспечения надлежащей эксплуатации

В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Значение *In* находится за пределами допустимого диапазона.

SecToDate

Команда SecToDate преобразует количество секунд, начиная с 00:00:00, 1 января 1970 г., в значение даты.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SecToDate	Преобразование секунд в дату	FUN	(@)SecToDate EN ENO In Out	Out:=SecToDate(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Секунды	Вход	Количество секунд, начиная с 00:00:00, 1 января 1970 г.	0...18446744073	Секунды	0
Out	Дата	Выход	Дата	Зависит от типа данных.	Год, месяц, день	---

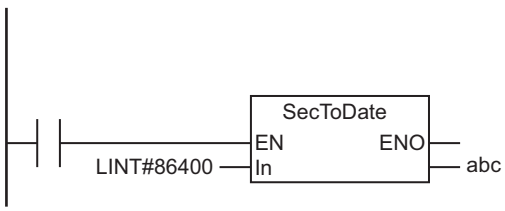
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In													OK							
Out																	OK			

Функция

Команда SecToDate преобразует переданное в *In* значение количества секунд, начиная с 00:00:0000:00:00, 1 января 1970 г., в значение даты. Данные после значения даты в полученном значении отбрасываются.

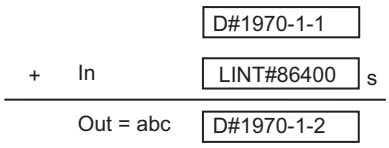
Ниже приведен пример для случая, когда *In* = LINT#86400.

LD



ST

```
abc:=SecToDate(LINT#86400);
```



Дополнительная информация

Для преобразования значения даты в значение количества секунд, начиная с 00:00:00, 1 января 1970 г., используйте команду *DateToSec* на стр. 2-713.

Меры предосторожности для обеспечения надлежащей эксплуатации

В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Значение *In* находится за пределами допустимого диапазона.

SecToTod

Команда SecToTod преобразует количество секунд, начиная с 00:00:00, в значение времени суток.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SecToTod	Преобразование секунд в время суток	FUN	(@)SecToTod EN ENO In Out	Out:=SecToTod(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Секунды	Вход	Количество секунд, начиная с 00:00:00	Зависит от типа данных.*1	Секунды	0
Out	Время суток	Выход	Время суток	Зависит от типа данных.	Час, минуты, секунды	---

*1. Отрицательные числа исключены.

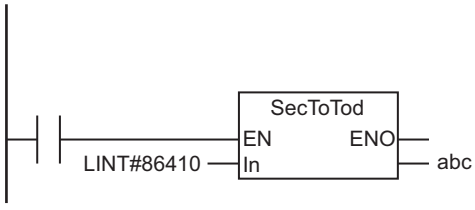
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In													OK							
Out																		OK		

Функция

Команда SecToTod преобразует переданное в *In* значение количества секунд, начиная с 00:00:00, в значение времени суток. Если значение в *In* равно 24 ч или больше, то *In* делится на 24 и остаток преобразуется в значение времени суток.

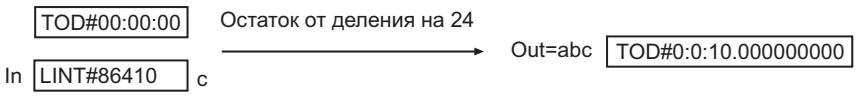
Ниже приведен пример для случая, когда *In* = LINT#86410.

LD



ST

```
abc:=SecToTod(LINT#86410);
```



Дополнительная информация

Для преобразования значения времени суток в количество секунд, начиная с 00:00:00, используйте команду *TodToSec* на стр. 2-715.

Меры предосторожности для обеспечения надлежащей эксплуатации

В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Значение *In* находится за пределами допустимого диапазона.

TimeToNanoSec

Команда TimeToNanoSec преобразует значение времени в наносекунды.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TimeToNanoSec	Преобразование времени в наносекунды	FUN	(@)TimeToNanoSec EN ENO In Out	Out:=TimeToNanoSec(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Время	Вход	Время	Зависит от типа данных.	нс	T#0s
Out	Наносекунды	Выход	Наносекунды	*1	нс	---

*1. -9223372036854775808...9223372036854775807

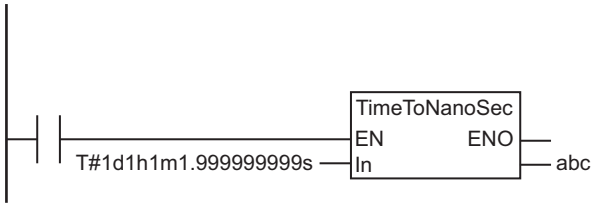
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																OK				
Out													OK							

Функция

Команда TimeToNanoSec преобразует значение времени *In* в количество наносекунд.

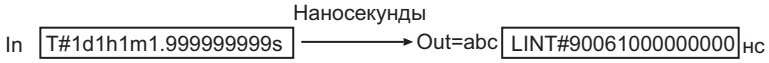
Ниже приведен пример для случая, когда *In* = T#1d1h1m1.999999999s.

LD



ST

```
abc:=TimeToNanoSec(T#1d1h1m1.999999999s);
```



Дополнительная информация

Для преобразования количества наносекунд в значение времени используйте команду *NanoSecToTime* на стр. 2-727.

TimeToSec

Команда TimeToSec преобразует значение времени в секунды.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TimeToSec	Преобразование времени в секунды	FUN	(@)TimeToSec EN ENO In Out	Out:=TimeToSec(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Время	Вход	Время	Зависит от типа данных.	нс	T#0s
Out	Секунды	Выход	Секунды	-9223372036... 9223372036	Секунды	---

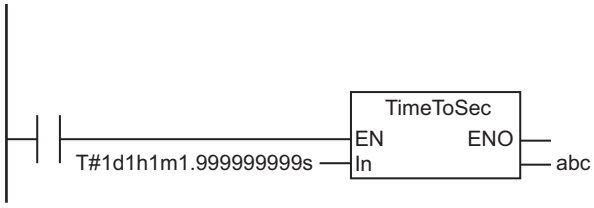
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																OK				
Out													OK							

Функция

Команда TimeToSec преобразует значение времени *In* в количество секунд. Данные после значения секунд в полученном значении отбрасываются.

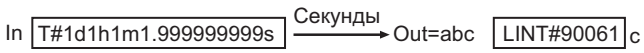
Ниже приведен пример для случая, когда *In* = T#1d1h1m1.999999999s.

LD



ST

```
abc:=TimeToSec(T#1d1h1m1.999999999s);
```



Дополнительная информация

Для преобразования количества секунд в значение времени используйте команду *SecToTime* на стр. 2-729.

Меры предосторожности для обеспечения надлежащей эксплуатации

Значение *In* измеряется в наносекундах. Значение *Out* измеряется в секундах.

NanoSecToTime

Команда NanoSecToTime преобразует количество наносекунд в значение времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NanoSecToTime	Преобразование наносекунд во время	FUN	(@)NanoSecToTime EN ENO In Out	Out:=NanoSecToTime(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Наносекунды	Вход	Наносекунды	*1	нс	0
Out	Время	Выход	Время	Зависит от типа данных.	нс	---

*1. -9223372036854775808...9223372036854775807

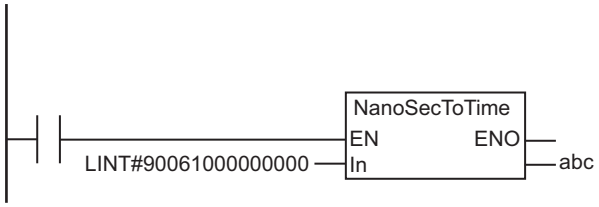
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In													OK							
Out																OK				

Функция

Команда NanoSecToTime преобразует значение количества наносекунд *In* в значение времени.

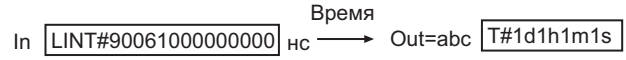
Ниже приведен пример для случая, когда *In* = LINT#90061000000000.

LD



ST

```
abc:=NanoSecToTime(LINT#90061000000000);
```



Дополнительная информация

Для преобразования значения времени в наносекунды используйте команду *TimeToNanoSec* на стр. 2-723.

SecToTime

Команда SecToTime преобразует количество секунд в значение времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SecToTime	Преобразование секунд во время	FUN	(@)SecToTime EN ENO In Out	Out:=SecToTime(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Секунды	Вход	Секунды	-9223372036... 9223372036	Секунды	0
Out	Время	Выход	Время	Зависит от типа данных.	нс	---

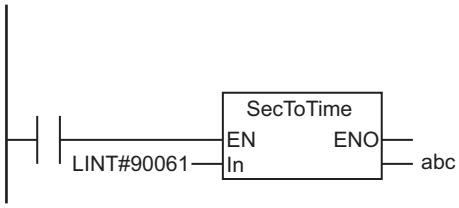
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In													OK							
Out																OK				

Функция

Команда SecToTime преобразует значение количества секунд *In* в значение времени.

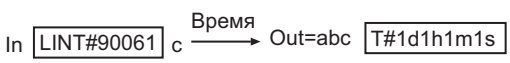
Ниже приведен пример для случая, когда *In* = LINT#90061.

LD



ST

```
abc:=SecToTime(LINT#90061);
```



Дополнительная информация

Для преобразования значения времени в секунды используйте команду *TimeToSec* на стр. 2-725.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Значение *In* измеряется в секундах. Значение *Out* измеряется в наносекундах.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  а) Значение *In* находится за пределами допустимого диапазона.

ChkLeapYear

Команда ChkLeapYear проверяет, является ли указанный год високосным.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ChkLeapYear	Проверка на високосный год	FUN	(@)ChkLeapYear EN ENO In Out	Out:=ChkLeapYear(In);

Переменные

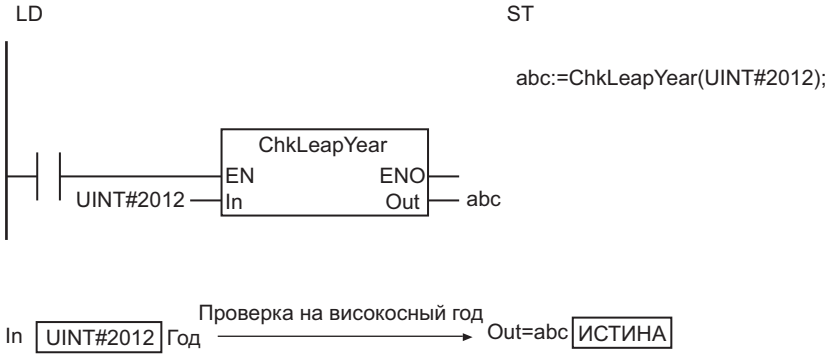
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Год	Вход	Год	1970...2554	Год	1970
Out	Результат	Выход	ИСТИНА: високосный год ЛОЖЬ: не високосный год	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In							OK													
Out	OK																			

Функция

Команда ChkLeapYear позволяет проверить, является ли год *In* високосным. Если это високосный год, то переменная *Out* = ИСТИНА. Если это не високосный год, то *Out* = ЛОЖЬ.

Ниже приведен пример для случая, когда *In* = UINT#2012.



Меры предосторожности для обеспечения надлежащей эксплуатации

Если значение *In* выходит за допустимый диапазон, ошибки не произойдет, а *Out* будет содержать недопустимое значение.

GetDaysOfMonth

Команда GetDaysOfMonth позволяет узнать количество дней в указанном месяце.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetDaysOfMonth	Получить количество дней в месяце	FUN	(@)GetDaysOfMonth EN Year Month ENO Out	Out:=GetDaysOfMonth(Year, Month);

Переменные

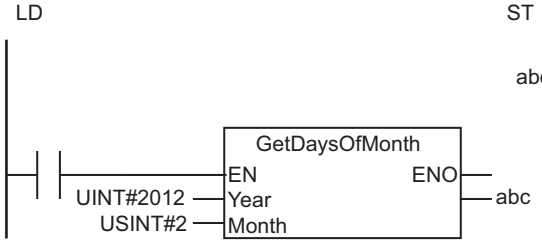
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Year	Год	Вход	Год	1970...2554	Год	1970
Month	Месяц		Месяц	1...12	Месяц	1
Out	Дни	Выход	Количество дней	28...31	Дни	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Year							ОК													
Month						ОК														
Out						ОК														

Функция

Команда GetDaysOfMonth возвращает количество дней в месяце *Month* года *Year*.

Ниже приведен пример, в котором *Year* = UINT#2012, а *Month* = USINT#2.



Меры предосторожности для обеспечения надлежащей эксплуатации

- Если значение *Year* выходит за допустимый диапазон, ошибки не произойдет, а *Out* будет содержать недопустимое значение.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Month* находится за пределами допустимого диапазона.

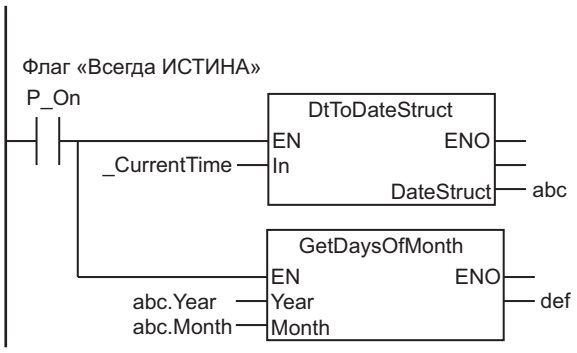
Пример программы

Будет рассмотрен пример использования команды для определения количества дней в текущем месяце.

Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	abc	_sDT	(Year:=0, Month:=0, Day:=0, Hour:=0, Min:=0, Sec:=0, NSec:=0)	Дата и время
	def	USINT	0	Количество дней в текущем месяце

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_CurrentTime	DATE_AND_TIME	☒	Системное время суток



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	abc	_sDT	(Year:=0, Month:=0, Day:=0, Hour:=0, Min:=0, Sec:=0, NSec:=0)	Дата и время

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	def	USINT	0	Количество дней в текущем месяце

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_CurrentTime	DATE_AND_TIME	☐	Системное время суток

```
DtToDateStruct(_CurrentTime, abc);
def:=GetDaysOfMonth(abc.Year, abc.Month);
```

DaysToMonth

Команда DaysToMonth вычисляет месяц на основании количества дней, начиная с 1 января.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DaysToMonth	Преобразование дней в месяц	FUN	(@)DaysToMonth EN ENO Year Out Days	Out:=DaysToMonth(Year, Days);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Year	Год	Вход	Год	1970...2554	Год	1970
Days	Дни		Количество дней, начиная с 1 января	1...365 1...366, когда *Year* — високосный год.	Дни	1
Out	Месяц	Выход	Месяц	1...12	Месяц	---

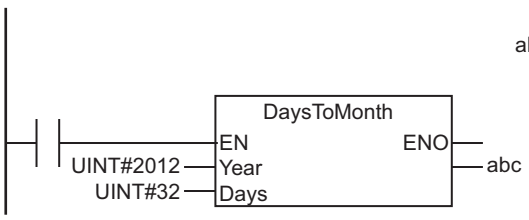
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Year							ОК													
Days							ОК													
Out						ОК														

Функция

Команда DaysToMonth вычисляет месяц на основании количества дней, заданного в параметре *Days*, начиная с 1 января года, заданного в параметре *Year*.

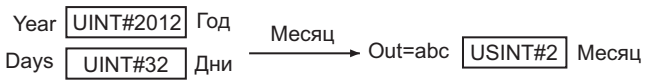
Ниже приведен пример, в котором *Year* = UINT#2012, а *Days* = UINT#32.

LD



ST

```
abc:=DaysToMonth(UINT#2012, UINT#32);
```



Меры предосторожности для обеспечения надлежащей эксплуатации

- Если значение *Year* выходит за допустимый диапазон, ошибки не произойдет, а *Out* будет содержать недопустимое значение.
- В указанном ниже случае произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Days* находится за пределами допустимого диапазона.

GetDayOfWeek

Команда GetDayOfWeek позволяет определить день недели для указанной даты (года, месяца и дня).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetDayOfWeek	Получить день недели	FUN	(@)GetDayOfWeek EN ENO In Out	Out:=GetDayOfWeek(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Год, месяц, день	Вход	Год, месяц, день	Зависит от типа данных.	Год, месяц, день	*1
Out	День недели	Выход	День недели	_MON, _TUE, _WED, _THU, _FRI, _SAT, _SUN	День недели	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																	OK		OK	
Out	Сведения о перечислителях перечислимого типа _eDAYOFWEEK см. в разделе *Функция* на стр. 2-738.																			

Функция

Команда GetDayOfWeek возвращает значение дня недели для даты (года, месяца и дня), указанной в *In*.

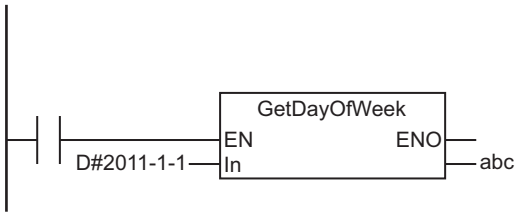
Для параметра *Out* используется перечислимый тип данных _eDAYOFWEEK. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_MON	Понедельник
_TUE	Вторник
_WED	Среда
_THU	Четверг
_FRI	Пятница

Перечислитель	Значение
_SAT	Суббота
_SUN	Воскресенье

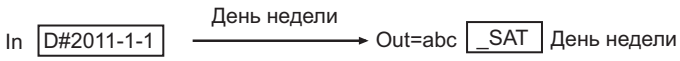
Ниже приведен пример для случая, когда *In* = D#2011-1-1.

LD



ST

```
abc:=GetDayOfWeek(D#2011-1-1);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

GetWeekOfYear

Команда GetWeekOfYear позволяет определить номер недели для указанной даты (года, месяца и дня).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetWeekOfYear	Получить номер недели	FUN	(@)GetWeekOfYear EN ENO In Out	Out:=GetWeekOfYear(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Год, месяц, день	Вход	Год, месяц, день	Зависит от типа данных.	Год, месяц, день	*1
Out	Week	Выход	Номер недели	1...54	Неделя	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																	OK		OK	
Out						OK														

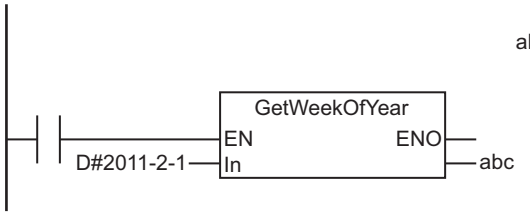
Функция

Команда GetWeekOfYear возвращает значение номера недели для даты (года, месяца и дня месяца), указанной в *In*.
Недели считаются с понедельника по воскресенье. Номер недели увеличивается на 1 при переходе с воскресенья на понедельник.

1 января всегда приходится на неделю 1.
Например, если 1 января — это четверг, то дни с 1 по 4 января (воскресенье) — это неделя 1, а дни с 5 января (понедельник) по 11 января (воскресенье) — это неделя 2.

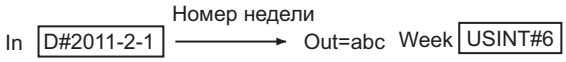
Ниже приведен пример для случая, когда *In* = D#2011-2-1.

LD



ST

```
abc:=GetWeekOfYear(D#2011-2-1);
```

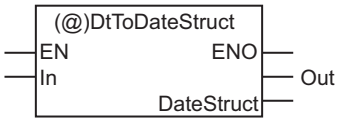


Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

DtToDateStruct

Команда DtToDateStruct преобразует дату и время в год, месяц, день, час, минуты, секунды и наносекунды.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DtToDateStruct	Разбивка даты и времени	FUN		Out:=DtToDateStruct(In, DateStruct);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Дата и время	Вход	Дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	DT#1970-1-1-0:0:0
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---
DateStruct	Дата и время		Дата и время в виде года, месяца, дня, часа, минут, секунд и наносекунд	---		

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																			ОК	
Out	ОК																			
DateStruct	Подробные сведения о структуре _sDT см. в разделе *Функция* на стр. 2-742.																			

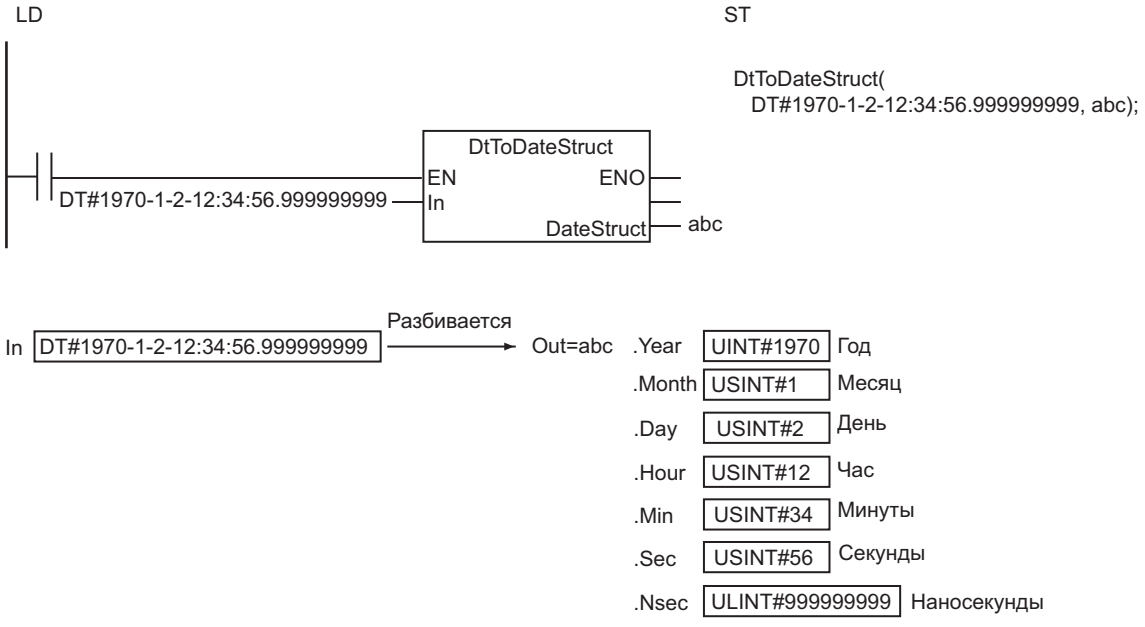
Функция

Эта команда преобразует переданное в *In* значение даты и времени в значения года, месяца, дня, часа, минут, секунд и наносекунд.

Для выходной переменной *DateStruct* используется структурный тип данных _sDT. Значения членов структуры приведены в таблице ниже.

Имя	Значение	Содержание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DateStruct	Дата и время	Дата и время в виде года, месяца, дня, часа, минут, секунд и наносекунд	_sDT	---	---	---
Year	Год	Год	UINT	1970...2554	Год	---
Month	Месяц	Месяц	USINT	1...12	Месяц	
Day	День месяца	День месяца	USINT	1...31	День месяца	
Hour	Час	Час	USINT	0...23	Час	
Min	Минуты	Минуты	USINT	0...59	Минуты	
Sec	Секунды	Секунды	USINT	0...59	Секунды	
Nsec	Наносекунды	Наносекунды	ULINT	0...999999999	Наносекунды	

В приведенном ниже примере *In* = DT#1970-1-2-12:34:56.999999999.



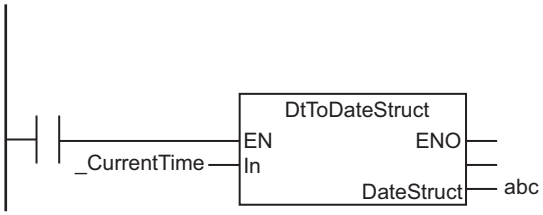
Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

Дополнительная информация

- Для объединения значений года, месяца, дня, часа, минут, секунд и наносекунд в значение даты и времени используйте команду *DateStructToDt* на стр. 2-745.
- Ниже показан пример определения текущего времени суток.

LD



ST

```
DtToDateStruct(_CurrentTime, abc);
```

Меры предосторожности для обеспечения надлежащей эксплуатации

При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.

DateStructToDt

Команда DateStructToDt объединяет значения года, месяца, дня, часа, минут, секунд и наносекунд в значение даты и времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DateStructToDt	Объединение времени	FUN	(@)DateStructToDt EN ENO In Out	Out:=DateStructToDt(In);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Дата и время	Вход	Дата и время в виде года, месяца, дня, часа, минут, секунд и наносекунд	---	---	---
Out	Дата и время	Выход	Дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	Подробные сведения о структуре _sDT см. в разделе *Функция* на стр. 2-745.																			
Out																			OK	

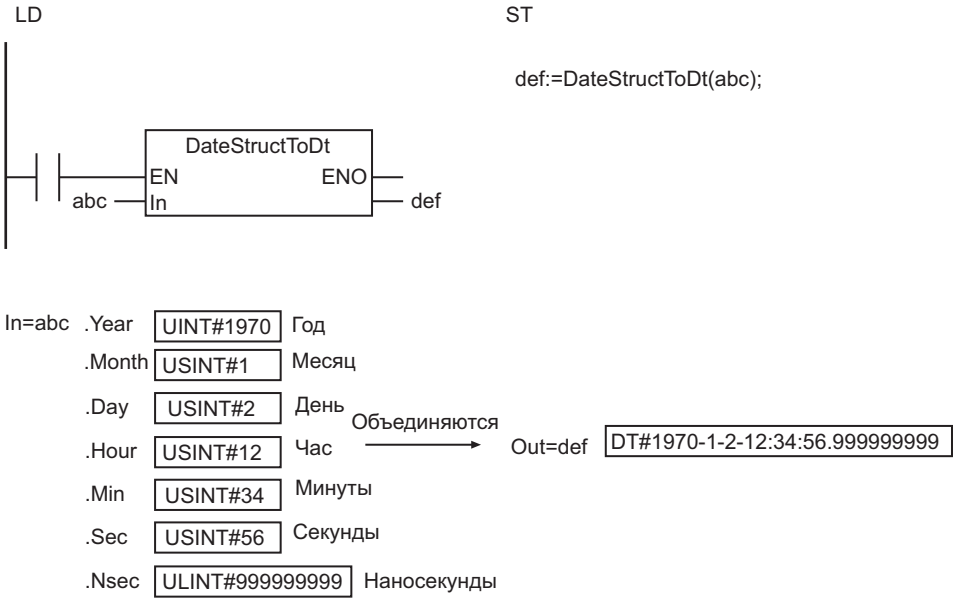
Функция

Команда DateStructToDt объединяет переданные в *In* значения года, месяца, дня, часа, минут, секунд и наносекунд в значение даты и времени.

Для переменной *In* используется структурный тип данных _sDT. Значения членов структуры приведены в таблице ниже.

Имя	Значение	Содержание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
In	Дата и время	Дата и время в виде года, месяца, дня, часа, минут, секунд и наносекунд	_sDT	---	---	---
Year	Год	Год	UINT	1970...2554	Год	1970
Month	Месяц	Месяц	USINT	1...12	Месяц	1
Day	День месяца	День месяца	USINT	1...31	День месяца	
Hour	Час	Час	USINT	0...23	Час	0
Min	Минуты	Минуты	USINT	0...59	Минуты	
Sec	Секунды	Секунды	USINT	0...59	Секунды	
Nsec	Наносекунды	Наносекунды	ULINT	0...999999999	Наносекунды	

В приведенном ниже примере члены переданной в *In* структуры имеют следующие значения: *Year* = UINT#1970, *Month* = USINT#1, *Day* = USINT#2, *Hour* = USINT#12, *Min* = USINT#34, *Sec* = USINT#56, а *Nsec* = ULINT#999999999.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CurrentTime	Системное время суток	DT	Текущее время суток по системным часам. Количество секунд начиная с 00:00:00 1 января 1970 г.

Дополнительная информация

Для разбивки значения даты и времени на год, месяц, день, час, минуты, секунды и наносекунды используйте команду *DtToDateStruct* на стр. 2-742.

Меры предосторожности для обеспечения надлежащей эксплуатации

В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.

- Значение члена структуры в *In* находится за пределами допустимого диапазона.
- Результат обработки выходит за диапазон допустимых значений *Out*.

TruncTime

Команда TruncTime усекает переменную типа TIME до заданной единицы измерения времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TruncTime	Усечение времени	FUN	(@)TruncTime EN ENO In Out Accuracy	Out:=TruncTime(In, Accuracy);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Время для усечения	Вход	Усекаемое время	Зависит от типа данных.	нс	T#0s
Accuracy	Наименьшая единица измерения после усечения		Наименьшая единица времени после усечения	_NANOSEC, _MICROSEC, _MILLISEC, _SEC	---	_NANOSEC
Out	Время после усечения	Выход	Время после усечения	Зависит от типа данных.	нс	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																ОК				
Accuracy	Сведения о перечислителях перечислимого типа _eSUBSEC см. в разделе *Функция* на стр. 2-748.																			
Out																ОК				

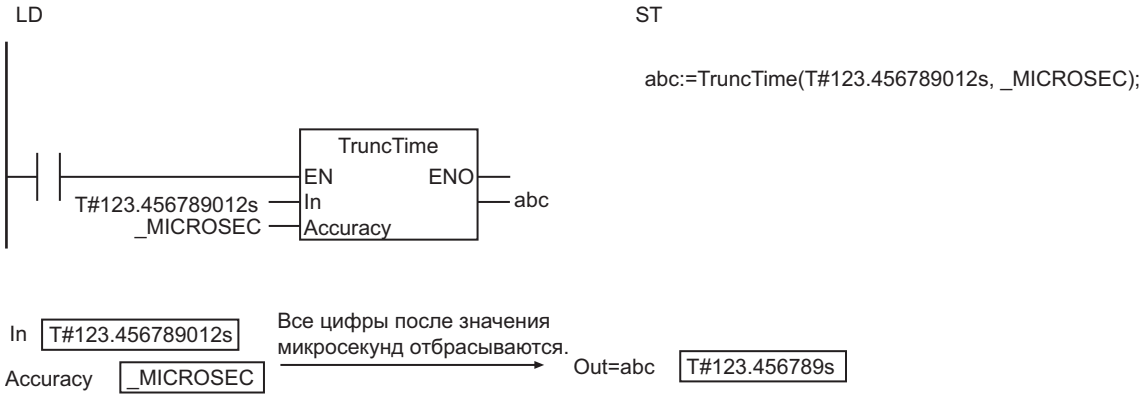
Функция

Команда TruncTime усекает значение времени *In* до единиц измерения времени, указанных в *Accuracy*. Результирующее значение времени после усечения сохраняется в переменную *Out*.

Для параметра *Accuracy* используется перечислимый тип данных _eSUBSEC. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_NANOSEC	Наносекунды
_MICROSEC	Микросекунды
_MILLISEC	Миллисекунды
_SEC	Секунды

Ниже приведен пример, в котором *In* = TIME#123.456789012s, а *Accuracy* = _MICROSEC.



Дополнительная информация

Прежде чем сравнивать две переменные времени (TIME) с помощью *EQ (=)* на стр. 2-108 или других команд, используйте эту команду для приведения двух переменных к одинаковой точности.

Пример программы

Ниже в качестве примера будет рассмотрена программа, которая определяет, превышает или равно ли время включения выхода датчика пороговому значению.
Возможны два режима работы: режим установки порогового уровня и режим выполнения. В приведенной ниже таблице описана работа в каждом из этих режимов.

Режим работы	Работа
Режим установки порога	Измеряется время включенного состояния выхода датчика, и полученное значение устанавливается в качестве порогового уровня.
Режим выполнения	Измеряется время включенного состояния выхода датчика, и полученное значение сравнивается с пороговым уровнем. Если время включения равно пороговому значению или превышает его, работа считается нормальной.

Время сравнивается с точностью до миллисекунд. Для усечения измеренного значения времени до единиц миллисекунд (знаки после единиц миллисекунд отбрасываются) используется команда TruncTime.
Текущий режим работы сохраняется в переменную *RecentMode*. Результат записывается в переменную *Result*.
Значение переменной *Result* равно ИСТИНА при нормальной работе и ЛОЖЬ — при ошибке.

Определения глобальных переменных

● Тип данных: Перечисление

Переменная	Перечислитель	Комментарий
Mode		Режим работы
SET	0	Настройка порога
EXEC	1	Выполнение

● Глобальные переменные

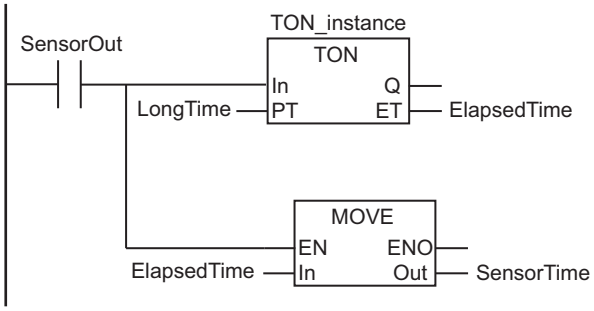
Переменная	Тип данных	Начальное значение	Комментарий
RecentMode	Mode	SET	Текущий режим работы

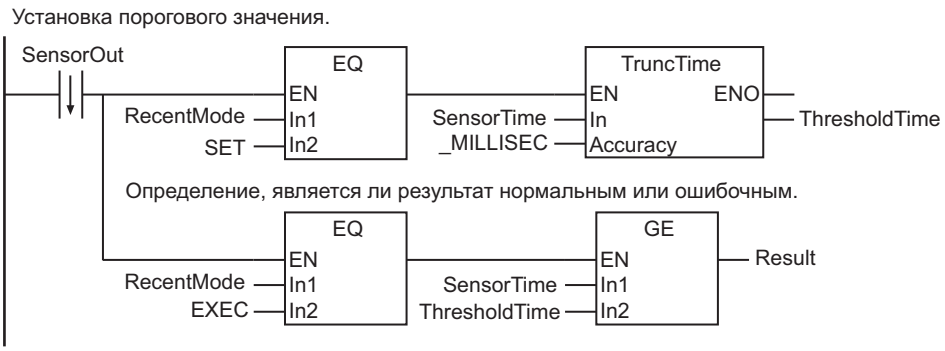
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	SensorOut	BOOL	ЛОЖЬ	Выход датчика
	ElapsedTime	TIME	T#0s	Истекшее время
	SensorTime	TIME	T#0s	Время, в течение которого включен датчик
	LongTime	TIME	T#1h	Время, которое с достаточным запасом превышает время включенного состояния датчика
	ThresholdTime	TIME	T#0s	Пороговое значение
	Результат	BOOL	ЛОЖЬ	Результат, ИСТИНА: норма; ЛОЖЬ: ошибка
	TON_instance	TON		

Внешние переменные	Переменная	Тип данных	Комментарий
	RecentMode	Mode	Текущий режим работы

Измерение времени, в течение которого выход датчика включен.





Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	SensorOut	BOOL	ЛОЖЬ	Выход датчика
	ElapsedTime	TIME	T#0s	Истекшее время
	SensorTime	TIME	T#0s	Время, в течение которого включен датчик
	LongTime	TIME	T#1h	Время, которое с достаточным запасом превышает время включенного состояния датчика
	SensorDone	BOOL	ЛОЖЬ	Флаг выключенного состояния выхода датчика
	ThresholdTime	TIME	T#0s	Пороговое значение
	Результат	BOOL	ЛОЖЬ	Результат, ИСТИНА: норма; ЛОЖЬ: ошибка
	TON_instance	TON		
	F_TRIG_instance	F_TRIG		

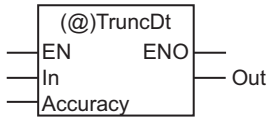
Внешние переменные	Переменная	Тип данных	Комментарий
	RecentMode	Mode	Текущий режим работы

```
// Выполнение команды TON.
TON_instance(
  In:=SensorOut, // Вход таймера
  PT:=LongTime, // Установка времени
  ET=>ElapsedTime); // Истекшее время

// Установка времени включения датчика равным истекшему времени TON.
IF (SensorOut=TRUE) THEN
  SensorTime:=ElapsedTime;
END_IF;

// Обнаружение момента выключения выхода датчика.
F_TRIG_instance(Clk:=SensorOut, Q=>SensorDone);
Result:=FALSE;
```

```
// Установка порога.
IF (SensorDone=TRUE AND RecentMode=SET) THEN
  ThresholdTime:=TruncTime(
    In :=SensorTime,
    Accuracy:=_MILLISEC); // Точность до миллисекунд.
// Определение результата: норма или ошибка.
ELSIF (SensorDone=TRUE AND RecentMode=EXEC) THEN
  IF (SensorTime >= ThresholdTime) THEN
    Result:=TRUE;
  END_IF;
END_IF;
```

TruncDt

Команда TruncDt усекает переменную типа DT до заданной единицы измерения времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TruncDt	Усечение даты и времени	FUN		Out:=TruncDt(In, Accuracy);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Дата и время для усечения	Вход	Усекаемые дата и время	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	DT#1970-1-1-0:0:0
Accuracy	Наименьшая единица измерения после усечения		Наименьшая единица времени после усечения	_NANOSEC, _MICROSEC, _MILLISEC, _SEC	---	_NANOSEC
Out	Дата и время после усечения	Выход	Дата и время после усечения	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																			OK	
Accuracy	Сведения о перечислителях перечислимого типа _eSUBSEC см. в разделе *Функция* на стр. 2-753.																			
Out																			OK	

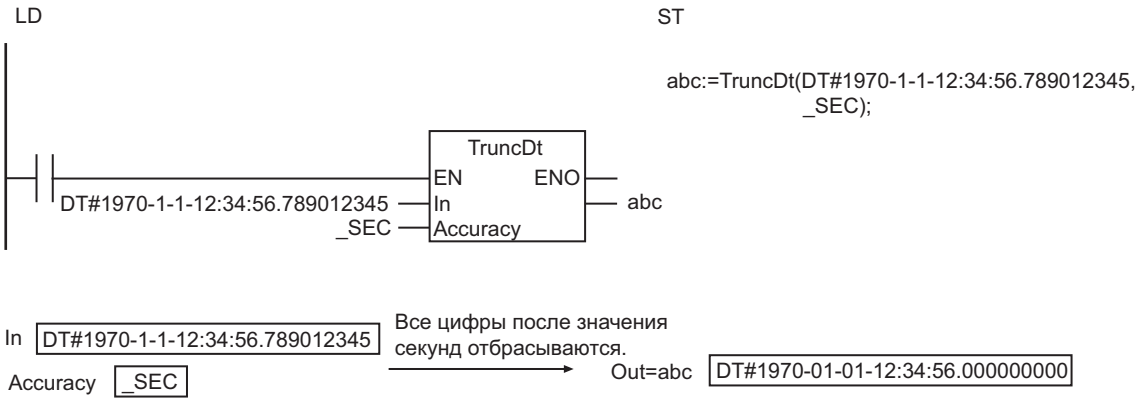
Функция

Команда TruncDt усекает переданное в *In* значение даты и времени до единиц измерения времени, указанных в *Accuracy*. Результирующее значение даты и времени после усечения сохраняется в переменную *Out*.

Для параметра *Accuracy* используется перечислимый тип данных _eSUBSEC. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_NANOSEC	Наносекунды
_MICROSEC	Микросекунды
_MILLISEC	Миллисекунды
_SEC	Секунды

Ниже приведен пример, в котором *In* = DT#1970-1-1-12:34:56.789012345, а *Accuracy* = _SEC.



Дополнительная информация

Прежде чем сравнивать две переменные даты и времени (DT) с помощью *EQ (=)* на стр. 2-108 или других команд, используйте эту команду для приведения двух переменных к одинаковой точности.

Пример программы

Ниже в качестве примера рассматривается программа, которая регистрирует дату и время, а также текущее напряжение в момент включения выхода датчика.
Дата и время регистрируются в миллисекундах.

Состояние выхода датчика сохраняется в переменную *SensorOut*, а величина напряжения — в переменную *Voltage*. Текущее значение даты и времени определяется с помощью команды GetTime.
Значения даты, времени и напряжения сохраняются по порядку в переменную *Stack* в виде структур *Recent*, членами которых являются дата, время и соответствующее напряжение.

Определения глобальных переменных

● Типы данных

Переменная	Тип данных	Комментарий
Record	STRUCT	Структура
DandT	DT	Дата и время
Voltage	REAL	Напряжение

● Глобальные переменные

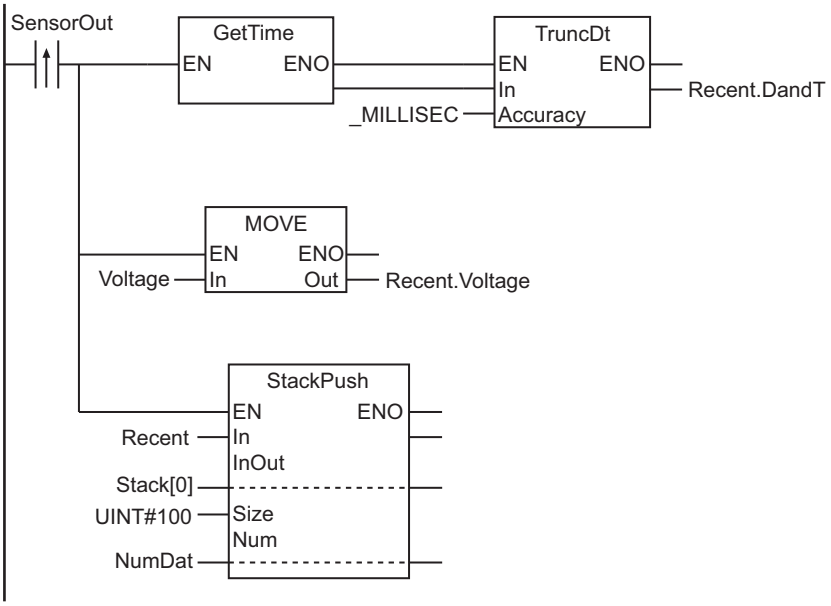
Переменная	Тип данных	Начальное значение	Комментарий
Recent	Record	(DandT:=DT#1970-1-1-0:0:0, Voltage:=0.0)	Текущее значение
Stack	ARRAY[0..99] OF Record	[100((DandT:=DT#1970-1-1-0:0:0, Voltage:=0.0))]	Стек

Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	SensorOut	BOOL	ЛОЖЬ	Выход датчика
	Voltage	REAL	0,0	Напряжение
	NumDat	UINT	UINT#0	Текущее количество хранимых данных

Внешние переменные	Переменная	Тип данных	Комментарий
	Recent	Record	Текущее значение
	Stack	ARRAY[0..99] OF Record	Стек

Запись даты, времени и напряжения



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	SensorOut	BOOL	ЛОЖЬ	Выход датчика

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Voltage	REAL	0,0	Напряжение
	NumDat	UINT	UINT#0	Текущее количество хранимых данных
	R_TRIG_instance	R_TRIG		

Внешние переменные	Переменная	Тип данных	Комментарий
	Recent	Record	Текущее значение
	Stack	ARRAY[0..99] OF Record	Стек

```
// Активировать условие выполнения при включении выхода датчика.
R_TRIG_instance(SensorOut, Trigger);

IF (Trigger=TRUE) THEN
  // Сохранять текущие дату и время с точностью до миллисекунды.
  Recent.DandT:=TruncDt(
    In :=GetTime(), // Получить дату и время.
    Accuracy:=_MILLISEC); // Точность до миллисекунд.

  // Получить текущее напряжение.
  Recent.Voltage:=Voltage;

  // Записать дату и время и напряжение в стек.
  StackPush(
    In :=Recent, // Дата, время и напряжение
    InOut:=Stack[0], // Массив стека
    Size :=UINT#100, // Количество элементов массива стека: 100
    Num :=NumDat); // Текущее количество хранимых данных
END_IF;
```

TruncTod

Команда TruncTod усекает переменную типа TOD до заданной единицы измерения времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TruncTod	Усечение времени суток	FUN	(@)TruncTod EN ENO In Out Accuracy	Out:=TruncTod(In, Accuracy);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.01 или более поздней и Sysmac Studio версии 1.02 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Время суток для усечения	Вход	Усекаемое время суток	Зависит от типа данных.	Час, минуты, секунды	TOD#0:0:0
Accuracy	Наименьшая единица измерения после усечения		Наименьшая единица времени после усечения	_NANOSEC, _MICROSEC, _MILLISEC, _SEC	---	_NANOSEC
Out	Время суток после усечения	Выход	Время суток после усечения	Зависит от типа данных.	Час, минуты, секунды	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																		OK		
Accuracy	Сведения о перечислителях перечислимого типа _eSUBSEC см. в разделе *Функция* на стр. 2-757.																			
Out																		OK		

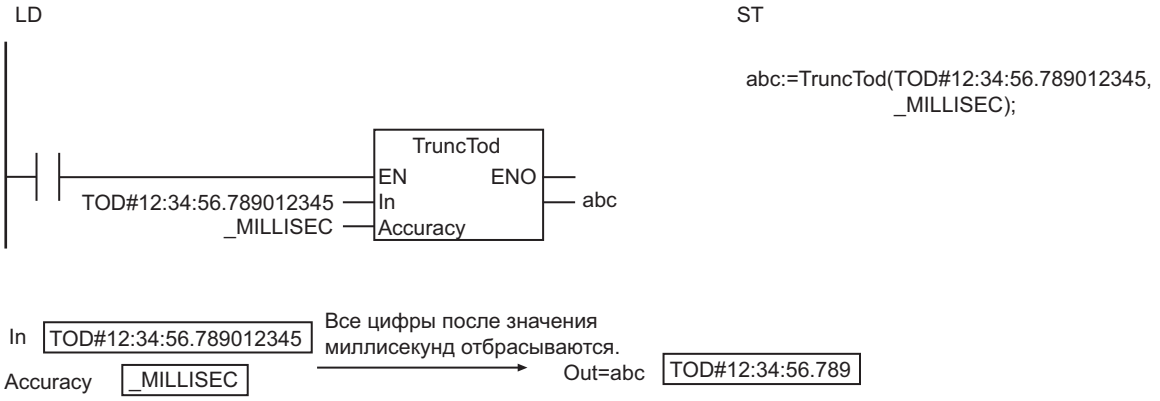
Функция

Команда TruncTod усекает значение времени суток *In* до единиц измерения времени, указанных в *Accuracy*. Результирующее значение времени суток после усечения сохраняется в переменную *Out*.

Для параметра *Accuracy* используется перечислимый тип данных _eSUBSEC. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_NANOSEC	Наносекунды
_MICROSEC	Микросекунды
_MILLISEC	Миллисекунды
_SEC	Секунды

Ниже приведен пример, в котором *In* = TOD#12:34:56.789012345, а *Accuracy* = _MILLISEC.



Дополнительная информация

Прежде чем сравнивать две переменные времени суток (TOD) с помощью *EQ (=)* на стр. 2-108 или других команд, используйте эту команду для приведения двух переменных к одинаковой точности.

Пример программы

Ниже в качестве примера рассматривается программа, которая регистрирует время суток, а также текущее напряжение в момент включения выхода датчика.
Время суток регистрируется в секундах.

Состояние выхода датчика сохраняется в переменную *SensorOut*, а величина напряжения — в переменную *Voltage*. Текущее время суток определяется с помощью команд GetTime и DT_TO_TOD.
Значения времени суток и напряжения сохраняются по порядку в переменную *Stack* в виде структур *Recent*, членами которых являются время суток и соответствующее напряжение.

Определения глобальных переменных

● Типы данных

Переменная	Тип данных	Комментарий
Record	STRUCT	Структура
TofD	TOD	Время суток
Voltage	REAL	Напряжение

● Глобальные переменные

Переменная	Тип данных	Начальное значение	Комментарий
Recent	Record	(TofD:=TOD#0:0:0, Voltage:=0.0)	Текущее значение

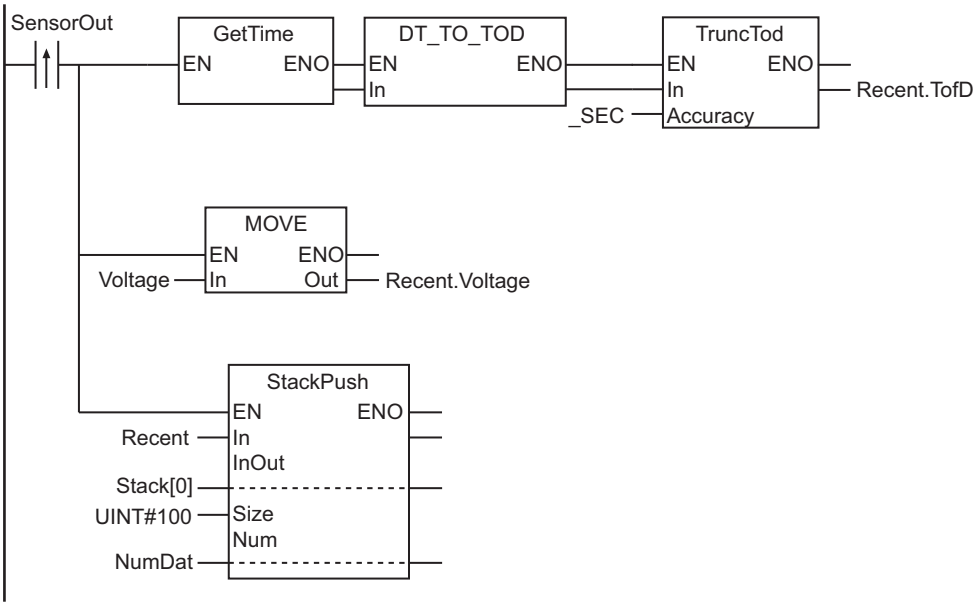
Переменная	Тип данных	Начальное значение	Комментарий
Stack	ARRAY[0..99] OF Record	[100((TofD:=TOD#0:0:0, Voltage:=0.0))]	Стек

Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	SensorOut	BOOL	ЛОЖЬ	Выход датчика
	Voltage	REAL	0,0	Напряжение
	NumDat	UINT	UINT#0	Текущее количество хранимых данных

Внешние переменные	Переменная	Тип данных	Комментарий
	Recent	Record	Текущее значение
	Stack	ARRAY[0..99] OF Record	Стек

Запись времени суток и напряжения



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	SensorOut	BOOL	ЛОЖЬ	Выход датчика
	TmpTod	TOD	TOD#0:0:0	Временная переменная
	Voltage	REAL	0,0	Напряжение
	NumDat	UINT	UINT#0	Текущее количество хранимых данных

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	R_TRIG_instance	R_TRIG		

Внешние переменные	Переменная	Тип данных	Комментарий
	Recent	Record	Текущее значение
	Stack	ARRAY[0..99] OF Record	Стек

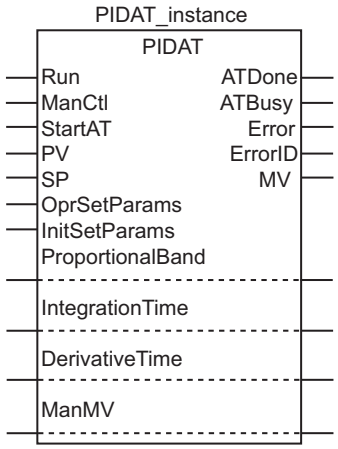
```
// Активировать условие выполнения при включении выхода датчика.
R_TRIG_instance(SensorOut, Trigger);

IF (Trigger=TRUE) THEN
  // Сохранять текущее время суток с точностью до секунд.
  TmpTod :=DT_TO_TOD(GetTime()); // Получить время суток.
  Recent.TofD:=TruncTod(
    In :=TmpTod,
    Accuracy:=_SEC); // Точность до секунд.

  // Получить текущее напряжение.
  Recent.Voltage:=Voltage;

  // Записать время суток и напряжение в стек.
  StackPush(
    In :=Recent, // Время суток и напряжение
    InOut:=Stack[0], // Массив стека
    Size :=UINT#100, // Количество элементов в массиве стека: 100
    Num :=NumDat); // Текущее количество хранимых данных
END_IF;
```

Команды для аналогового регулирования

Команда	Имя	Стр.
PIDAT	ПИД-регулятор с автонастройкой	стр. 2-762
PIDAT_HeatCool	ПИД-регулятор нагрева/охлаждения с автонастройкой	стр. 2-796
TimeProportionalOut	Выход широтно-импульсного регулирования	стр. 2-839
LimitAlarm_**	Группа сигнализации аварии выхода за верхний/нижний предел	стр. 2-860
LimitAlarmDv_**	Группа сигнализации аварии верхнего/нижнего отклонения	стр. 2-865
LimitAlarmDvStbySeq_**	Группа сигнализации аварии верхнего/нижнего отклонения с контролем последовательности	стр. 2-871
ScaleTrans	Изменение масштаба	стр. 2-891
AC_StepProgram	Поэтапная программа	стр. 2-894

PIDAT

Команда PIDAT реализует ПИД-регулятор с автонастройкой (2-ПИД-регулятор с фильтром уставки).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
PIDAT	ПИД-регулятор с автонастройкой	FB		PIDAT_instance(Run, ManCtl, StartAT, PV, SP, OprSetParams, InitSetParams, ProportionalBand, IntegrationTime, DerivativeTime, ManMV, ATDone, ATBusy, Error, ErrorID, MV);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Run	Условие выполнения	Вход	ИСТИНА: выполнить ЛОЖЬ: остановить	Зависит от типа данных.	---	ЛОЖЬ
ManCtl	Ручное/автоматическое управление		ИСТИНА: работа в режиме ручного управления ЛОЖЬ: работа в автоматическом режиме			
StartAT	Условие выполнения автонастройки		ИСТИНА: выполнить ЛОЖЬ: отмена			
PV	Регулируемая величина		Регулируемый технологический параметр	*1		0
SP	Уставка		Заданное значение			
OprSetParams	Рабочие параметры		Параметры, задаваемые во время работы	---		---
InitSetParams	Начальные параметры		Начальные параметры			

	Значение	Вход/выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Proportional Band	Зона пропорциональности	Вход-выход	Зона пропорциональности	0,01...1000,00	% от полн. диап.	---
IntegrationTime	Постоянная времени интегрирования		Постоянная времени интегрирования Чем больше это значение, тем меньше управляющее воздействие по интегралу. При значении 0 интегральное звено отключено.	T#0.0000 с ... T#10000.0000 с*2	с	
DerivativeTime	Постоянная времени дифференцирования		Постоянная времени дифференцирования Чем больше это значение, тем больше управляющее воздействие по производной. При значении 0 дифференциальное звено отключено.	T#0.0000 с ... T#10000.0000 с*2		
ManMV	Ручное управляющее воздействие		Управляющее воздействие при управлении вручную	-320...320	%	
ATDone	Нормальное завершение автонастройки	Выход	ИСТИНА: нормальное завершение ЛОЖЬ:*3	Зависит от типа данных.	---	---
ATBusy	Выполнение автонастройки		ИСТИНА: выполняется автонастройка ЛОЖЬ: автонастройка не выполняется			
MV	Управляющее воздействие		Управляющее воздействие	-320...320	%	

*1. От значения *InitSetParams.RngLowLmt* (нижняя граница входного диапазона) до значения *InitSetParams.RngUpLmt* (верхняя граница входного диапазона)
*2. Значение усекается до четырех знаков после запятой.
*3. Значение ЛОЖЬ означает, что команда завершена с ошибкой, что в данный момент выполняется ПИД-регулирование без автонастройки или что в данный момент ПИД-регулирование не выполняется.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Run	OK																			
ManCtl	OK																			
StartAT	OK																			
PV														OK						
SP														OK						

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
OprSetParams	Подробные сведения о структуре _sOPR_SET_PARAMS см. в разделе *Описание структурного типа* на стр. 2-764.																			
InitSetParams	Подробные сведения о структуре _sINIT_SET_PARAMS см. в разделе *Описание структурного типа* на стр. 2-764.																			
Proportional Band														ОК						
IntegrationTime																ОК				
DerivativeTime																ОК				
ManMV														ОК						
ATDone	ОК																			
ATBusy	ОК																			
MV														ОК						

Функция

Команда PIDAT осуществляет пропорционально-интегрально-дифференциальное регулирование (ПИД-регулирование) и формирует управляющее воздействие для регулятора температуры или другого устройства.
ПИД-регулирование запускается, когда значение переменной *Run* (условие выполнения) меняется на ИСТИНА. Пока значение *Run* = ИСТИНА, циклически повторяется следующая рабочая последовательность: считывается текущее значение регулируемой величины *PV*, выполняются расчеты ПИД-регулятора («ПИД-обработка») и на выход подается рассчитанное управляющее воздействие *MV*.
Когда значение переменной *Run* меняется на ЛОЖЬ, ПИД-регулирование останавливается.

Поддерживается функция автоматической настройки для автоматического определения оптимальных значений констант ПИД-регулятора.
Автоматическая настройка констант ПИД-регулятора производится, когда значение переменной *StartAT* (условие выполнения автонастройки) меняется на ИСТИНА.

Описание структурного типа

Для переменной **OprSetParams** (рабочие параметры) используется структурный тип данных _sOPR_SET_PARAMS. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
OprSetParams	Рабочие параметры	Параметры, задаваемые во время работы.	_sOPR_SET_PARAMS	---	---	---

Имя		Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
	MVLowLmt	Нижний предел управляющего воздействия	Нижнее предельное значение переменной *MV*.	REAL	-320...320*1	%	0
	MVUpLmt	Верхний предел управляющего воздействия	Верхнее предельное значение переменной *MV*.	REAL			100
	ManResetVal	Значение ручного сброса	Значение *MV* при нулевом отклонении в случае пропорционального регулирования.	REAL	-320...320		0
	MVTrackSw	Переключатель отслеживания управляющего воздействия (MV)	ИСТИНА: ВКЛ ЛОЖЬ: ВЫКЛ	BOOL	Зависит от типа данных.	---	ЛОЖЬ
	MVTrackVal	Отслеживаемое значение управляющего воздействия (MV)	Значение, устанавливаемое в переменной *MV* во время отслеживания управляющего воздействия.	REAL	-320...320	%	0
	StopMV	Управляющее воздействие при остановке	Значение, устанавливаемое в переменной *MV*, когда выполнение команды остановлено.	REAL			
	ErrorMV	Управляющее воздействие при ошибке	Значение, устанавливаемое в переменной *MV*, когда возникает ошибка.	REAL			
	Alpha	Параметр α 2-ПИД регулятора	Коэффициент α фильтра уставки. Если это значение равно 0, фильтр уставки отключен.	REAL	0,00...1,00	---	0,65
	ATCalcGain	Коэффициент влияния автонастройки	Коэффициент регулировки по результатам автонастройки. При более высоких значениях более высоким приоритетом обладает стабильность. При меньших значениях более приоритетной является скорость реакции.	REAL	0,1...10,0		1,0
	ATHystrs	Гистерезис автонастройки	Гистерезис предельного цикла.	REAL		% от полн. диап.	0,2

*1. Значение *MVLowLmt* должно быть меньше значения *MVUpLmt*.

Для переменной **InitSetParams** (начальные параметры) используется структурный тип данных _sINIT_SET_PARAMS. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
InitSetParams	Начальные параметры	Начальные параметры.	_sINIT_SET_PARAMS	---	---	---
SampTime	Период измерения	Период выполнения ПИД-обработки.	TIME	T#0.0001 с ... #100.0000 с	с	T#0.1 с
RngLowLmt	Нижняя граница входного диапазона	Нижнее предельное значение переменных *PV* и *SP*.	REAL	-32000...32000*1	---	0
RngUpLmt	Верхняя граница входного диапазона	Верхнее предельное значение переменных *PV* и *SP*.	REAL			100
DirOpr	Направление действия	ИСТИНА: прямое действие. ЛОЖЬ: обратное действие.	BOOL	Зависит от типа данных.		ЛОЖЬ

*1. Значение *RngLowLmt* должно быть меньше значения *RngUpLmt*.

Назначение переменных

Ниже поясняется назначение переменных, которые используются в данной команде.

● Run (условие выполнения)

Эта переменная играет роль условия выполнения команды.
ПИД-регулирование производится, когда она содержит значение ИСТИНА. Когда значение этой переменной меняется на ЛОЖЬ, ПИД-регулирование прекращается.

● ManCtl (ручное/автоматическое управление)

Эта команда может выполняться в одном из двух режимов: в режиме ручного управления или в автоматическом режиме.
Значение переменной *ManCtl* определяет используемый режим.

Значение *ManCtl*	Режим работы	Значение *MV*
ИСТИНА	Ручное управление	Значение *ManMV* (ПИД-регулирование не производится)
ЛОЖЬ	Автоматическое управление	Значение, рассчитанное ПИД-регулятором

● StartAT (условие выполнения автонастройки)

Эта переменная играет роль условия выполнения автоматической настройки постоянных ПИД-регулятора.
Если переменная *StartAT* уже содержит значение ИСТИНА в момент перехода переменной *Run* в состояние ИСТИНА, автоматическая настройка констант выполняется при запуске ПИД-регулятора.
Если же значение *StartAT* меняется на ИСТИНА во время ПИД-регулирования (т. е. когда значение *Run* = ИСТИНА), автоматическая настройка выполняется во время ПИД-регулирования.

И в том и в другом случае автонастройка отменяется, если значение *StartAT* меняется на ЛОЖЬ во время автонастройки.
Информацию о функции автоматической настройки см. в разделе *Автоматическая настройка* на стр. 2-780.

● PV (регулируемая величина)

Это регулируемый технологический параметр управляемой системы.

● SP (уставка)

Это заданное значение технологического параметра управляемой системы.

● MVLowLmt (нижний предел MV) и MVUpLmt (верхний предел MV)

С помощью этих переменных можно ограничить диапазон значений переменной *MV*.
В переменных *MVLowLmt* и *MVUpLmt* задаются соответственно нижнее и верхнее предельные значения управляющего воздействия (MV).
Значение *MVLowLmt* всегда должно быть меньше значения *MVUpLmt*.

Управляющее воздействие в результате ПИД-обработки	Значение *MV*
Меньше значения *MVLowLmt*	*MVLowLmt*
Между значениями *MVLowLmt* и *MVUpLmt* включительно	Управляющее воздействие в результате ПИД-обработки
Больше значения *MVUpLmt*	*MVUpLmt*

В отношении значений *StopMV* (упр. воздействие при остановке), *ErrorMV* (упр. воздействие при ошибке) и *ManMV* (ручное упр. воздействие), которые устанавливаются в переменную *MV* в соответствующих случаях, ограничение не применяется.

Значения *MVLowLmt* и *MVUpLmt* можно изменять, даже если команда в данный момент не находится в состоянии «автоматическая настройка при работе в автоматическом режиме».
Однако если *MVLowLmt* и *MVUpLmt* будут изменены в сторону расширения диапазона во время работы команды, будет выдано такое же значение *MV*, как в последнем цикле измерения, а затем оно плавно («безударно») изменится.
Частое изменение значений *MVLowLmt* и *MVUpLmt* может отрицательно повлиять на качество регулирования и сделать характеристики регулирования неудовлетворительными.
Прежде чем повторно изменять параметр *MVLowLmt* или *MVUpLmt* во время работы, убедитесь, что это не ухудшит качество регулирования.

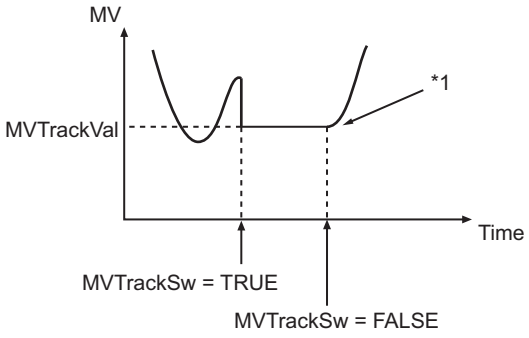
● ManResetVal (значение ручного сброса)

Это значение переменной *MV* при пропорциональном управлении, когда отклонение (т. е. разница между *PV* и *SP*) равно 0.
Значение параметра *ManResetVal* определяет местоположение зоны пропорциональности.
При использовании интегрального звена значение ручного сброса игнорируется. Следовательно, заданное значение *ManResetVal* действует, когда значение *IntegrationTime* = 0.

● MVTrackSw (переключатель отслеживания управляющего воздействия)

Функция отслеживания управляющего воздействия (MV) записывает в переменную *MV* внешнее входное значение (которое называется отслеживаемым значением управляющего воздействия), когда ПИД-регулятор работает в автоматическом режиме.
Отслеживание управляющего воздействия производится, когда значение переменной *MVTrackSw* = ИСТИНА.

Когда значение переменной *MVTrackSw* снова становится равно ЛОЖЬ, переменная *MV* принимает значение переменной *MVTrackVal* в текущем измерительном цикле, а начиная со следующего измерительного цикла в нее записывается результат вычислений ПИД-регулятора.
Это позволяет избежать резкого изменения значения *MV*.



*1. *MV* принимает значение *MVTrackVal*.

● MVTrackVal (отслеживаемое значение управляющего воздействия)

Это значение, которое записывается в переменную *MV* во время работы функции отслеживания управляющего воздействия (MV).
Значение *MVTrackVal* ограничено значениями параметров *MVLowLmt* и *MVUpLmt*.

● StopMV (управляющее воздействие при остановке)

Это значение, которое переменная *MV* принимает, когда значение переменной *Run* меняется на ЛОЖЬ (т. е. когда выполнение этой команды прекращается).

● ErrorMV (управляющее воздействие при ошибке)

Это значение, которое переменная *MV* принимает, когда возникает ошибка (т. е. когда значение *Error* = ИСТИНА).
Если значение *ErrorMV* находится за пределами допустимого диапазона (-320...320), значение *MV* будет равно 0 при возникновении ошибки.

● Alpha (параметр α 2-ПИД регулятора)

Этот параметр задает значение коэффициента для фильтра уставки.
Дополнительные сведения см. в разделе *2-ПИД-регулятор с фильтром уставки* на стр. 2-778.
В общем случае, для параметра *Alpha* следует установить значение 0,65.

● ATCalcGain (коэффициент влияния автонастройки)

Эта переменная позволяет задать коэффициент, с которым постоянные ПИД-регулятора, рассчитанные во время автонастройки, применяются в ПИД-регуляторе.
При значении 1,00 применяются непосредственно значения, полученные в ходе автонастройки.
Для достижения большей стабильности значение *ATCalcGain* следует увеличить, а для повышения скорости реакции — уменьшить.

● ATHystrs (гистерезис автонастройки)

Эта переменная позволяет задать гистерезис, используемый в предельном цикле для выполнения автонастройки.

Чем меньше значение *ATHystrs*, тем точнее выполняется автонастройка. Однако в том случае, если регулируемая величина нестабильна и автонастройка не выполняется должным образом, это значение следует увеличить.
Дополнительные сведения см. в разделе *Автоматическая настройка* на стр. 2-780.

● SampTime (период измерения)

Эта переменная позволяет задать минимальный период, с которым работает ПИД-регулятор.
Дополнительные сведения см. в разделе *Время выполнения ПИД-регулирования* на стр. 2-782.
Обработка в ПИД-регуляторе не выполняется, если время, прошедшее с момента последнего выполнения обработки, меньше времени *SampTime*.

● RngLowLmt (нижняя граница входного диапазона) и RngUpLmt (верхняя граница входного диапазона)

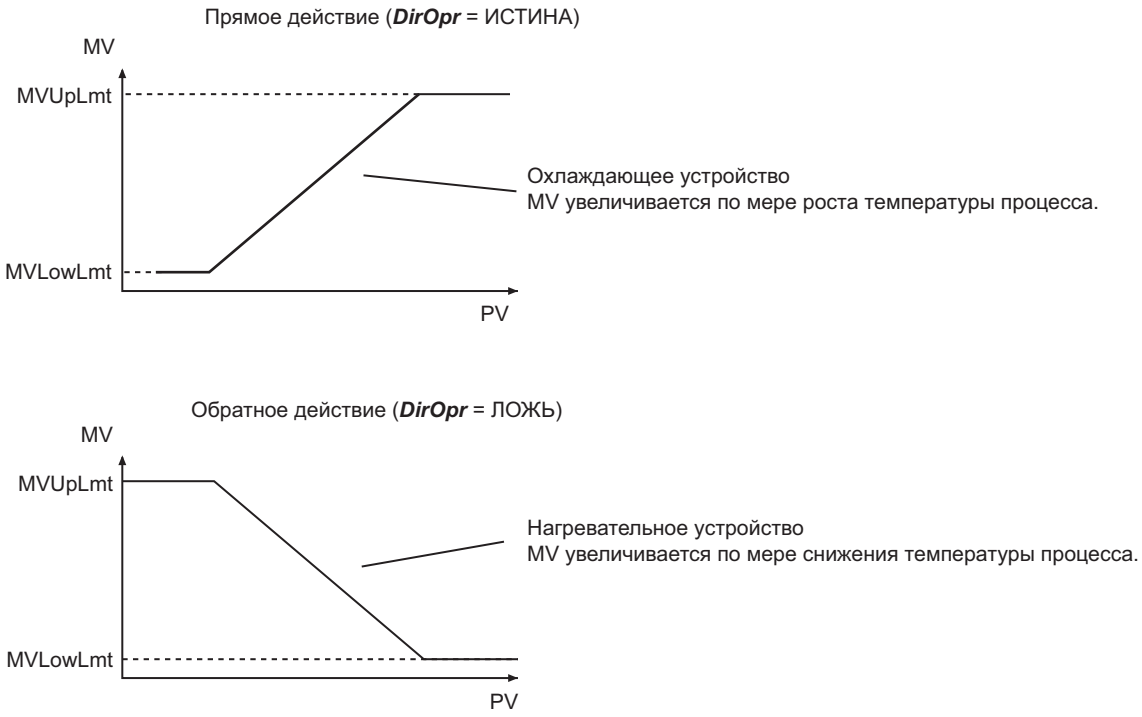
С помощью этих переменных можно задать верхнее и нижнее предельные значения для переменных *PV* и *SP*.
Если значение параметра, подключенного к переменной *PV* или *SP*, выйдет за эти пределы, произойдет ошибка.
Значение *RngLowLmt* всегда должно быть меньше значения *RngUpLmt*.

● DirOpr (направление действия)

Эта переменная указывает, увеличивается или уменьшается значение *MV* при изменении значения *PV*.
Это называется соответственно прямым действием и обратным действием.

Значение *DirOpr*.	Значение	Значение *MV*
ИСТИНА	Прямое действие	Увеличивается при увеличении значения *PV*.
ЛОЖЬ	Обратное действие	Уменьшается при увеличении значения *PV*.

Ниже поясняется разница между прямым и обратным действием для регулирования температуры.
Прямое действие используется для формирования управляющего воздействия, подаваемого на охлаждающее устройство. Это значит, что чем выше регулируемая температура, тем больше величина управляющего воздействия, подаваемого на охлаждающее устройство. Обратное действие используется для формирования управляющего воздействия, подаваемого на нагревательное устройство. В данном случае чем ниже регулируемая температура, тем больше величина управляющего воздействия, подаваемого на нагревательное устройство.



● ProportionalBand (зона пропорциональности)

Это одна из трех постоянных ПИД-регулятора. Дополнительные сведения см. в разделе *Пропорциональное управляющее воздействие (P)* на стр. 2-773.
Чем больше значение *ProportionalBand*, тем больше величина статической ошибки. При слишком маленьком значении параметра ProportionalBand могут возникать автоколебания.

● IntegrationTime (постоянная времени интегрирования)

Это одна из трех постоянных ПИД-регулятора. Дополнительные сведения см. в разделе *Управляющее воздействие по интегралу (I)* на стр. 2-775.
Чем больше значение переменной *IntegrationTime*, тем меньше управляющее воздействие по интегралу.

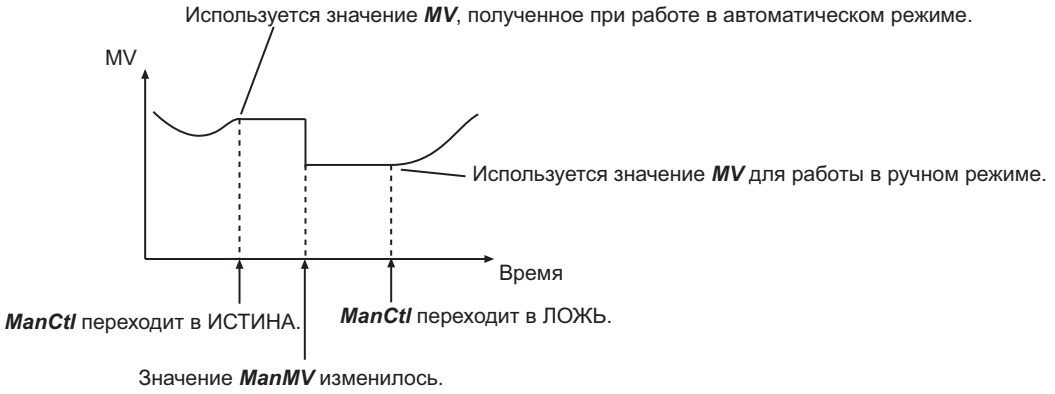
● DerivativeTime (постоянная времени дифференцирования)

Это одна из трех постоянных ПИД-регулятора. Дополнительные сведения см. в разделе *Управляющее воздействие по производной (D)* на стр. 2-776.
Чем больше значение переменной *DerivativeTime*, тем больше управляющее воздействие по производной.

● ManMV (ручное управляющее воздействие)

Переменная *MV* принимает это значение во время работы в режиме ручного управления (когда *ManCtl* = ИСТИНА).
Однако непосредственно после переключения ПИД-регулятора из автоматического режима в ручной режим по-прежнему применяется значение *MV*, полученное при работе в автоматическом режиме.
Переменная *MV* принимает значение *ManMV*, только когда значение *ManMV* изменяется после переключения ПИД-регулятора в ручной режим.
После переключения из ручного в автоматический режим продолжает применяться значение *MV* для работы в ручном режиме.

Значение *ManMV* не обязательно должно находиться в диапазоне между *MVLowLmt* и *MVUpLmt*.



● ATDone (нормальное завершение автонастройки)

Этот флаг указывает, что автоматическая настройка была завершена нормально.
Он переходит в состояние ИСТИНА, когда автонастройка завершается нормально, и остается в состоянии ИСТИНА до тех пор, пока значение переменной *StartAT* = ИСТИНА.

В указанных ниже случаях этот флаг находится в состоянии ЛОЖЬ.

- Операция автонастройки завершилась с ошибкой.
- Автонастройка выполняется в данный момент (т. е. пока значение *ATBusy* = ИСТИНА).
- В данный момент выполняется ПИД-регулирование без автонастройки.
- В данный момент ПИД-регулирование не выполняется (т. е. значение *Run* = ЛОЖЬ).
- Значение переменной *StartAT* = ЛОЖЬ.

● ATBusy (выполнение автонастройки)

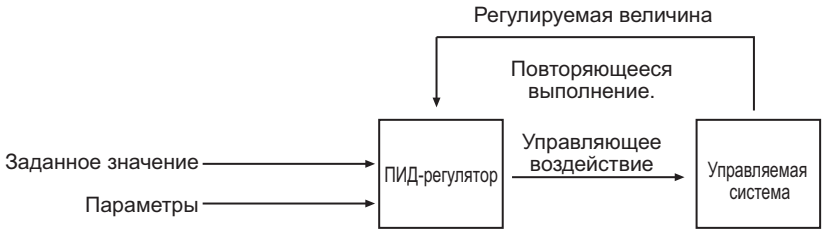
Этот флаг указывает на то, что в данный момент выполняется автоматическая настройка.
Он находится в состоянии ИСТИНА, пока производится автонастройка. Все остальное время он находится в состоянии ЛОЖЬ.

● MV (управляющее воздействие)

Это управляющее воздействие, которое подается на исполнительный механизм в управляемой системе.

Введение в ПИД-регулирование

Пропорционально-интегрально-дифференциальное регулирование (ПИД-регулирование) — это метод управления с обратной связью, который состоит в непрерывном (или циклическом) измерении регулируемой величины управляемой системы и вычислении такого управляющего воздействия, которое приводит регулируемую величину к заданному значению.
Таким образом, данная команда выдает управляющее воздействие, которое определяется на основании следующих входных данных: регулируемая величина (параметр технологического процесса), заданное значение (уставка) и параметры для выполнения расчетов.
ПИД-регулятор циклически измеряет текущее значение регулируемой величины, производит вычисления и подает на выход рассчитанное управляющее воздействие, в результате чего регулируемая величина приводится к заданному значению.

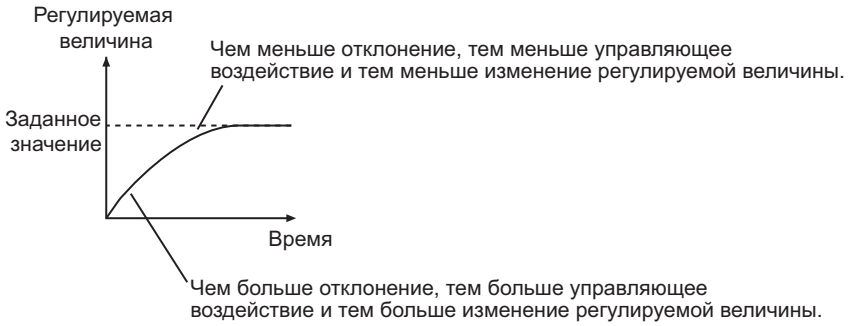


Пропорциональное (P), интегральное (I) и дифференциальное (D) управление

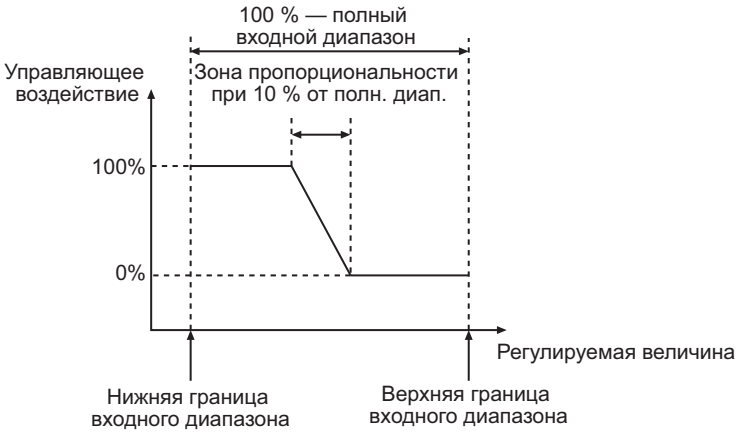
ПИД-регулятор сочетает в себе три вида управления: пропорциональное (P), интегральное (I) и дифференциальное (D).

● Пропорциональное управляющее воздействие (P)

Пропорциональное воздействие (или воздействие по отклонению) — вид управления, при котором абсолютное значение управляющего воздействия (MV) изменяется пропорционально величине отклонения регулируемой величины от заданного значения.
На рисунке ниже показан характер изменения регулируемой величины управляемой системы.

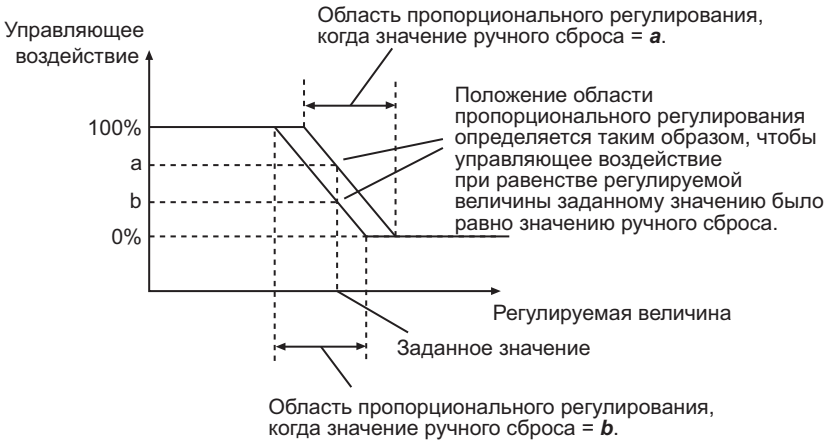


Одним из настраиваемых параметров пропорционального управления является зона пропорциональности.
Зона пропорциональности определяется как диапазон значений регулируемой величины, в котором применяется пропорциональное управляющее воздействие. Если текущее значение регулируемой величины находится за пределами зоны пропорциональности, для управляющего воздействия устанавливается значение 100 % или 0 %.
Зона пропорциональности выражается в процентах от входного диапазона, в котором выполняется пропорциональное управление («% от полн. диапаз»). На приведенном ниже рисунке зона пропорциональности установлена равной 10 % от полного диапазона.

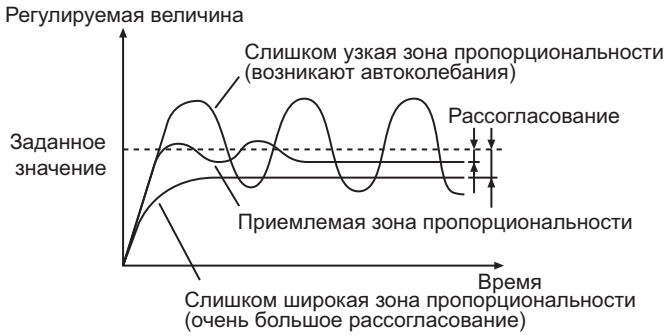


Еще одним параметром при пропорциональном управлении является значение ручного сброса. Значение ручного сброса — это значение управляющего воздействия при нулевом отклонении регулируемой величины от заданного значения.
Фактически значение ручного сброса определяет положение участка пропорционального управления на графике зависимости управляющего воздействия от текущего значения регулируемой величины.

Взаимосвязь между значением ручного сброса и областью пропорционального регулирования показана на рисунке ниже.
Положение области пропорционального регулирования определяется таким образом, чтобы управляющее воздействие при равенстве регулируемой величины заданному значению было равно значению ручного сброса.



При неподходящем значении ручного сброса нулевое отклонение никогда не будет достигнуто. Это остаточное отклонение регулируемой величины от заданного значения также называют статической ошибкой регулирования.
Для уменьшения статической ошибки можно сузить зону пропорциональности. Однако если зона пропорциональности будет слишком узкой, регулируемая величина, достигнув заданного значения, «проскочит» его и продолжит изменяться. Это явление называют перерегулированием. Система также может утратить устойчивость, и могут наблюдаться периодические колебания регулируемой величины относительно заданного значения. Такое явление называют автоколебаниями.



● Управляющее воздействие по интегралу (I)

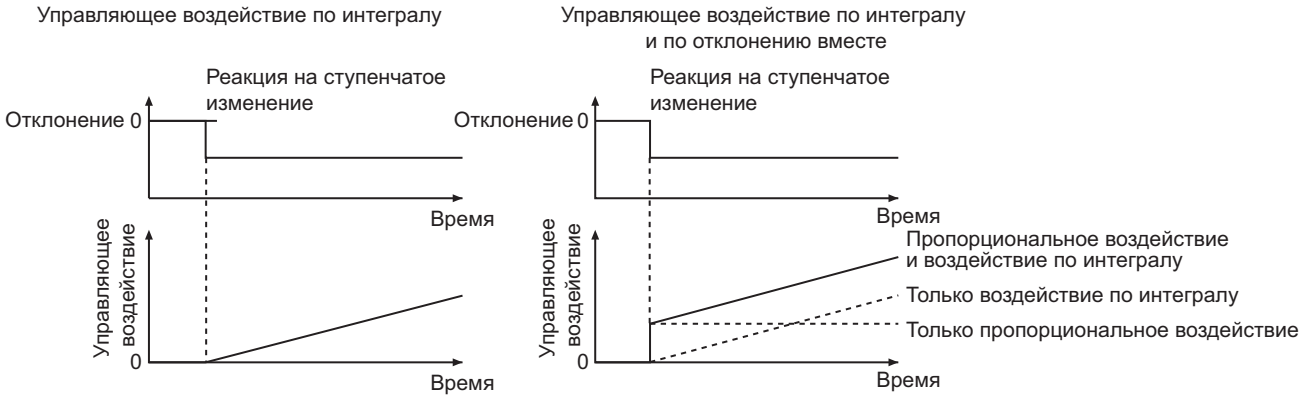
При использовании только пропорционального звена практически невозможно достичь нулевого отклонения регулируемой величины от заданного значения в установившемся режиме. Для этого требуется очень точная настройка зоны пропорциональности и значения ручного сброса. Кроме того, величина статической ошибки зависит от величины возмущения, поэтому настройку необходимо часто повторять.

Для устранения указанных выше недостатков пропорционального регулирования в сочетании с пропорциональным звеном используют интегральное звено, обеспечивающее управляющее воздействие по интегралу.

Интегральное звено рассчитывает интеграл по времени от отклонения регулируемой величины от заданного значения и изменяет абсолютное значение управляющего воздействия пропорционально результату.

При использовании интегрального звена значение ручного сброса игнорируется.

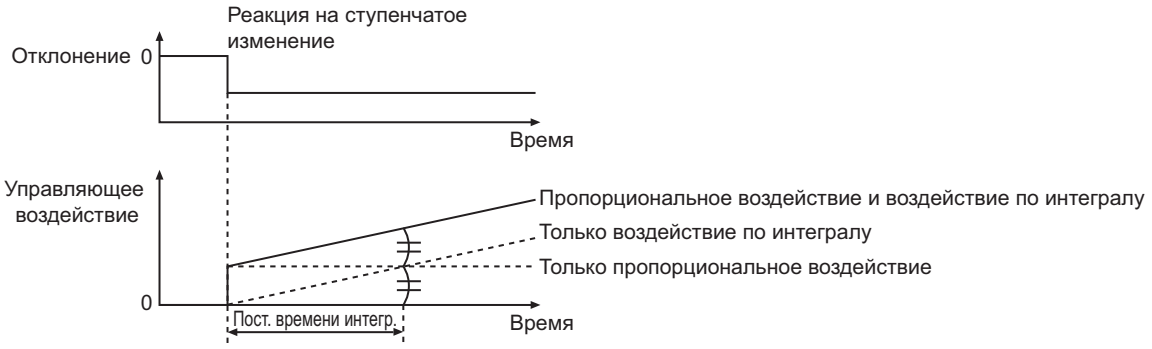
На приведенном ниже рисунке слева показано, как изменяется управляющее воздействие на выходе интегрального звена в ответ на ступенчатое изменение отклонения. Там же справа показано, как изменяется управляющее воздействие, когда интегральное и пропорциональное звенья используются вместе.



Одним из параметров при интегральном управлении является постоянная времени интегрирования.

Она определяется как время, за которое управляющее воздействие на выходе интегрального звена становится равно управляющему воздействию на выходе пропорционального звена при ступенчатом изменении отклонения регулируемой величины от заданного значения.

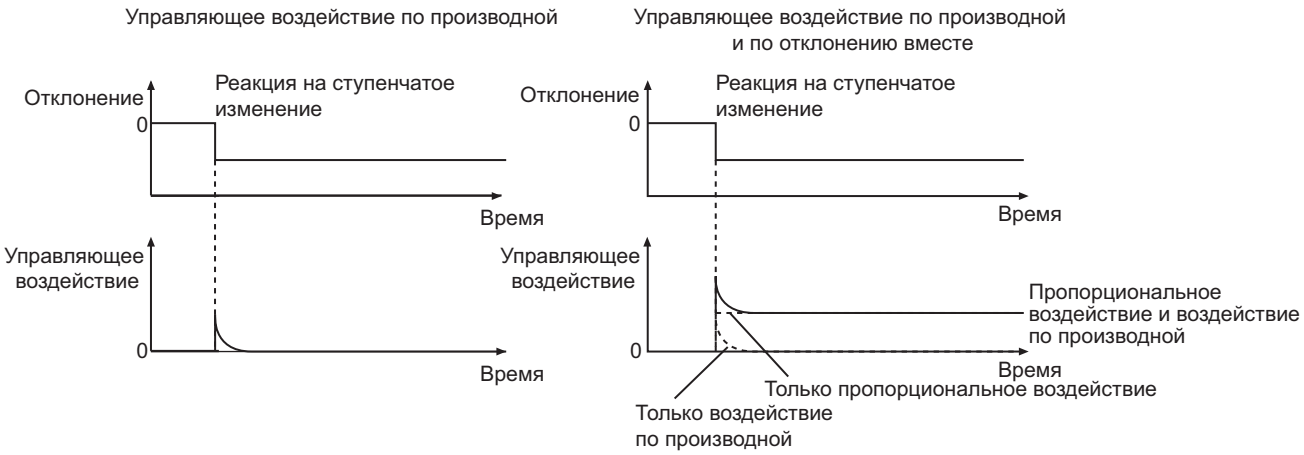
Чем меньше постоянная времени интегрирования, тем сильнее интегральное звено воздействует на регулируемую величину. Уменьшая постоянную времени интегрирования, можно ускорить достижение нулевого значения отклонения, однако это также может привести к возникновению автоколебаний.



● Управляющее воздействие по производной (D)

При пропорционально-интегральном регулировании величина отклонения со временем сводится к нулю и регулируемая величина становится равна заданному значению.
Однако если под действием возмущений регулируемая величина изменяется очень быстро, системе может потребоваться значительное время для восстановления первоначального состояния.
Чтобы регулируемая величина быстро возвращалась к заданному значению при наличии возмущений, используют управляющее воздействие по производной, то есть дифференциальное звено.
Дифференциальное звено рассчитывает производную по времени от отклонения и изменяет абсолютное значение управляющего воздействия пропорционально результату. Другими словами, чем больше изменяется регулируемая величина за единицу времени, тем больше абсолютное значение управляющего воздействия на выходе дифференциального звена.

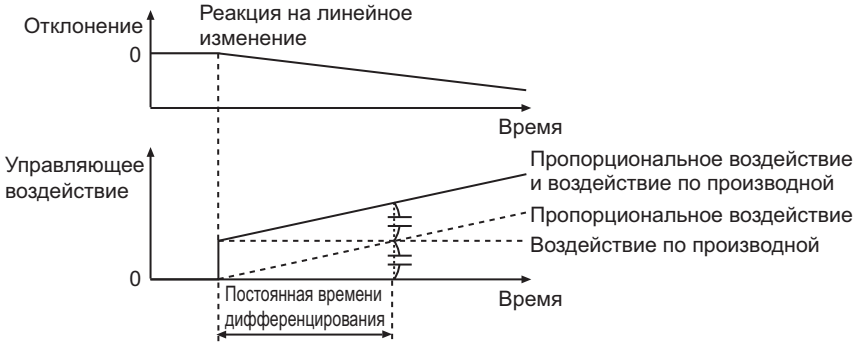
На рисунке ниже показано, как изменяется управляющее воздействие при дифференциальном регулировании в случае ступенчатого изменения величины отклонения. Там же показано изменение управляющего воздействия при совместном использовании дифференциального и пропорционального звеньев.



Одним из параметров при дифференциальном управлении является постоянная времени дифференцирования. Она определяется как время, за которое управляющее воздействие на выходе дифференциального звена становится равно управляющему воздействию на выходе пропорционального звена при линейном изменении отклонения регулируемой величины от заданного значения.

Чем больше постоянная времени дифференцирования, тем сильнее дифференциальное звено воздействует на регулируемую величину. Увеличивая постоянную времени

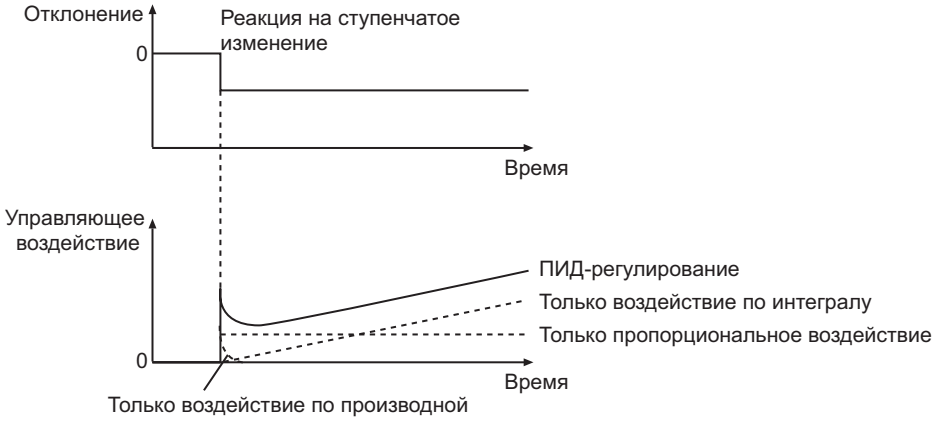
дифференцирования, можно ускорить реакцию регулятора на возмущения, однако это также может привести к возникновению автоколебаний.

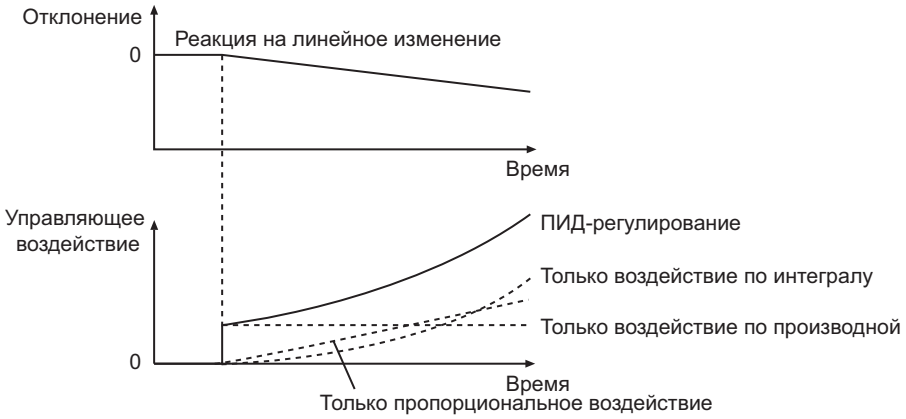


● ПИД-регулирование

Управляющее воздействие на выходе ПИД-регулятора можно представить как сумму управляющих воздействий на выходах пропорционального (П), интегрального (И) и дифференциального (Д) звеньев этого регулятора.

На рисунке ниже приведены кривые изменения управляющего воздействия на выходе ПИДрегулятора при ступенчатом и линейном изменении величины отклонения.





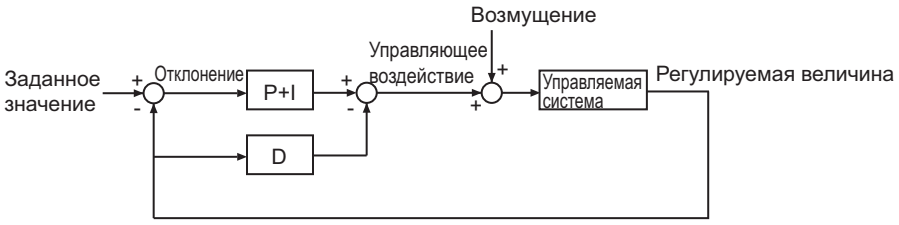
2-ПИД-регулятор с фильтром уставки

Для осуществления ПИД-регулирования необходимо настроить три основных параметра: зону пропорциональности, постоянную времени интегрирования и постоянную времени дифференцирования. Эти параметры называются постоянными (или константами) ПИД-регулятора.
От значений постоянных ПИД-регулятора зависят две его характеристики, которые указаны ниже.

- Переходная характеристика по задающему воздействию: способность регулятора реагировать на изменения в заданном значении.
- Переходная характеристика по возмущающему воздействию: способность регулятора компенсировать возмущения, приводящие к значительному изменению регулируемой величины, и приводить регулируемую величину к заданному значению.

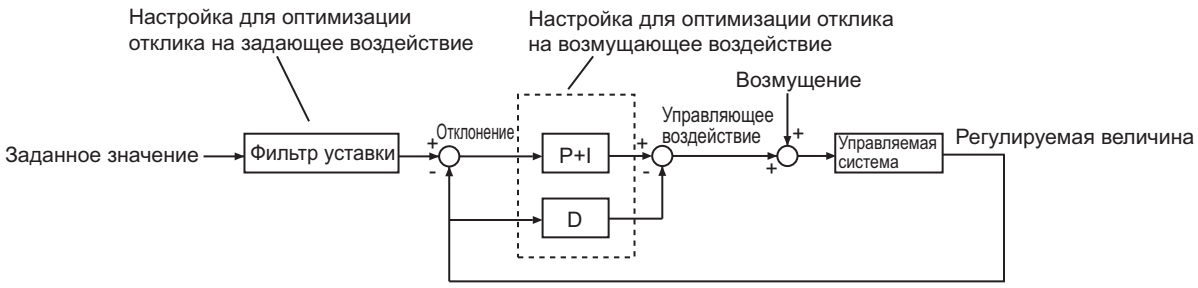
На рисунке ниже представлена базовая функциональная схема ПИД-регулятора.

Как видно из функциональной схемы, задающее и возмущающее воздействия вводятся в ПИД-регулятор через разные входы. Это затрудняет поиск значений постоянных ПИД-регулятора, которые были бы одинаково оптимальными как для переходной характеристики по задающему воздействию, так и для переходной характеристики по возмущающему воздействию. Другими словами, если постоянные ПИД-регулятора оптимизированы для переходной характеристики по задающему воздействию, то скорость реагирования на возмущения может быть низкой. А если значения постоянных оптимизированы с точки зрения переходной характеристики по возмущающему воздействию, в системе могут возникать автоколебания.

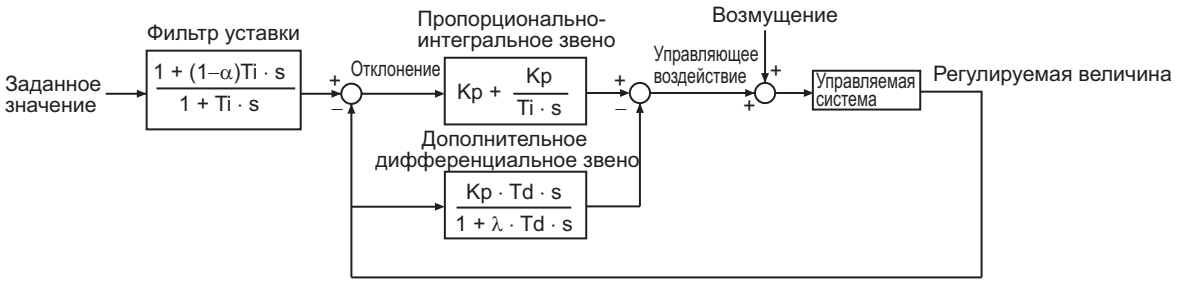


Для получения оптимальных переходных характеристик как по задающему, так и по возмущающему воздействиям используется технология ПИД-регулирования под названием «2-PID» (или «2-ПИД-регулирование»).
Цифра «2» в названии технологии "2-PID" означает, что для оптимизации отклика на изменение заданного значения и для оптимизации отклика на возмущающее воздействие используются отдельные (разные) параметры.

Функциональная схема ПИД-регулятора такого типа изображена на рисунке ниже.
По сравнению с обычным ПИД-регулятором здесь добавлен фильтр задающего воздействия (или «фильтр уставки»), который содержит настраиваемый параметр.
Постоянные ПИД-регулятора настраиваются для обеспечения максимальной скорости реагирования на возмущения. Фильтр уставки корректирует задающее воздействие для получения оптимальной скорости отклика регулятора на изменение заданного значения при заданных значениях ПИД-констант.
Значения постоянных ПИД-регулятора и значение параметра фильтра уставки можно настраивать раздельно, что позволяет добиваться высокого быстродействия одновременно по задающему и возмущающему воздействиям.



Ниже приведены формулы, которые используются в функциональных блоках этой команды.
Передаточная характеристика фильтра уставки настраивается с использованием постоянной времени интегрирования и параметра α 2-ПИД регулятора.
Оптимальное значение коэффициента α: 0,65. Как правило, это значение изменять не требуется. Чем меньше значение α, тем меньше фильтр уставки влияет на работу регулятора.



Kp: коэффициент передачи (коэффициент пропорционального действия)
Ti: постоянная времени интегрирования
Td: постоянная времени дифференцирования
s: оператор Лапласа
α: параметр 2-ПИД-регулирования
λ: коэффициент неполной производной

Запуск ПИД-регулирования

Для выполнения этой команды необходимо использовать подходящие значения постоянных ПИД-регулятора. Возможны два варианта запуска ПИД-регулирования. Выбор соответствующего варианта зависит от того, известны ли к моменту запуска оптимальные значения постоянных ПИД-регулятора.

Значения постоянных ПИД-регулятора можно изменять во время его работы. Также во время работы можно выполнить автоматическую настройку. Для запуска автонастройки во время работы необходимо поменять значение переменной *StartAT* на ИСТИНА.

● Когда подходящие ПИД-константы неизвестны

В начале работы необходимо выполнить автонастройку для определения подходящих значений постоянных ПИД-регулятора.
Для этого при переходе входа *Run* в состояние ИСТИНА вход *StartAT* должен находиться в состоянии ИСТИНА.
В этом случае сначала выполняется автонастройка, а затем запускается ПИД-регулирование с использованием найденных значений ПИД-констант.

● Когда подходящие ПИД-константы известны

Присвойте оптимальные значения постоянных ПИД-регулятора переменным *ProportionalBand*, *IntegrationTime* и *DerivativeTime*, а затем переведите вход *Run* в состояние ИСТИНА.
ProportionalBand, *IntegrationTime* и *DerivativeTime* являются входными-выходными переменными,
поэтому в них нельзя передавать константы в качестве входных параметров. Определите подходящие переменные и присвойте значения входным параметрам.

Управляющее воздействие в разных состояниях регулятора

Управляющее воздействие (*MV*) определяется в соответствии с текущим состоянием регулятора, что отражено в таблице ниже.

<table>
<tr><th rowspan="2">Состояние регулятора</th><th colspan="5">Значение переменной</th><th rowspan="2">MV (управляющее воздействие)</th></tr>
<tr><th>ManCtl (ручное/автоматическое управление)</th><th>Run (условие выполнения)</th><th>Error (завершение с ошибкой)</th><th>MVTrackSw (переключатель отслеживания MV)</th><th>ATBusy (выполнение автонастройки)</th></tr>
<tr><td>Завершение с ошибкой</td><td rowspan="6">ЛОЖЬ</td><td rowspan="5">ИСТИНА</td><td>ИСТИНА</td><td>---</td><td rowspan="3">ЛОЖЬ</td><td>ErrorMV
(управляющее воздействие при ошибке)</td></tr>
<tr><td>Во время работы в автоматическом режиме (отслеживание MV)</td><td rowspan="4">ЛОЖЬ</td><td>ИСТИНА</td><td>MVTrackVal
(отслеживаемое значение MV)</td></tr>
<tr><td>Во время работы в автоматическом режиме (автонастройка)</td><td rowspan="2">ЛОЖЬ</td><td>ИСТИНА</td><td>Значение многократно изменяется в диапазоне между нижним и верхним предельными значениями MV.</td></tr>
<tr><td>Во время работы в автоматическом режиме (без автонастройки)</td><td rowspan="3">ЛОЖЬ</td><td>Значение, рассчитанное по текущим значениям ПИД-констант.</td></tr>
<tr><td>Выполнение команды остановлено</td><td>ЛОЖЬ</td><td rowspan="2">---</td><td rowspan="2">---</td><td>StopMV
(управляющее воздействие при остановке)</td></tr>
<tr><td>Работа в режиме ручного управления</td><td>ИСТИНА</td><td>---</td><td>ManMV
(ручное управляющее воздействие)</td></tr>
</table>

Автоматическая настройка

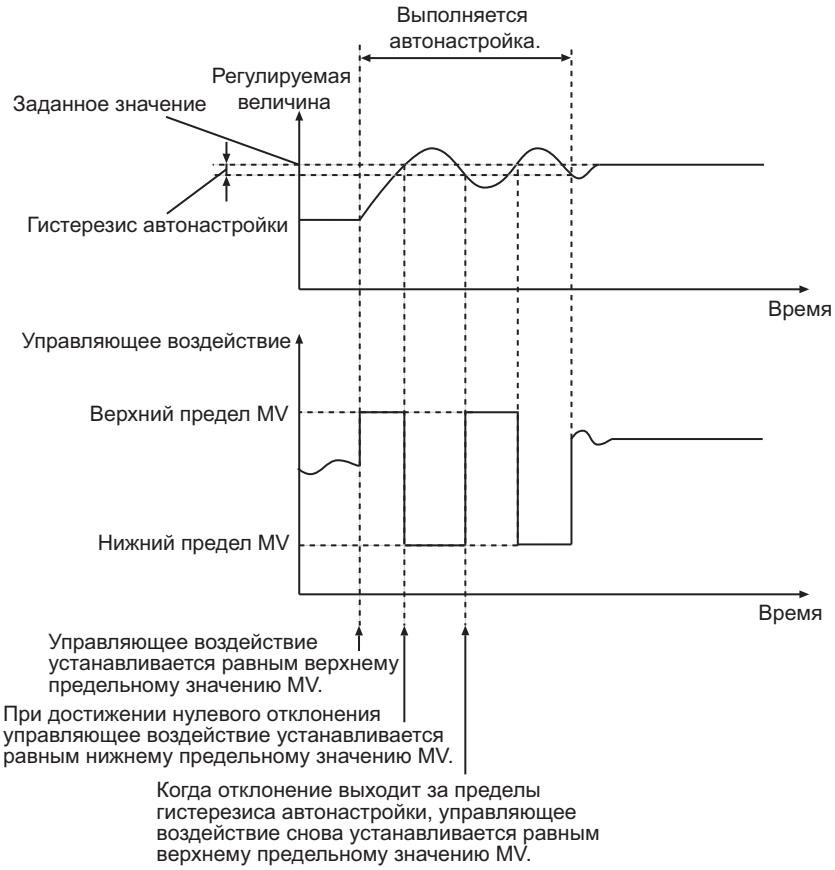
Параметр α 2-ПИД регулятора не требуется настраивать часто. Таким образом, основными параметрами, которые настраиваются для этой команды, являются постоянные ПИД-регулятора. Команда PIDAT поддерживает автоматическую настройку постоянных ПИД-регулятора.

Для автонастройки используется метод предельного цикла.

Суть метода предельного цикла состоит в том, что на выходе регулятора временно устанавливаются верхнее и нижнее предельные значения управляющего воздействия, регистрируются соответствующие изменения регулируемой величины и на основе этого определяются оптимальные значения постоянных ПИД-регулятора.

Если при запуске автонастройки регулируемая величина меньше заданного значения, управляющее воздействие сначала устанавливается равным верхнему предельному значению. При достижении нулевого отклонения управляющее воздействие устанавливается равным нижнему предельному значению. Когда отклонение выходит за пределы гистерезиса автонастройки, управляющее воздействие снова устанавливается равным верхнему предельному значению. Для расчета оптимальных ПИД-констант этот процесс выполняется дважды.

Если при запуске автонастройки регулируемая величина больше заданного значения, управляющее воздействие сначала устанавливается равным нижнему предельному значению. Затем производится расчет оптимальных значений ПИД-констант с использованием описанной выше процедуры.



Автонастройка выполняется во время ПИД-регулирования, если значение *StartAT* меняется на ИСТИНА во время ПИД-регулирования (т. е. когда значение *Run* = ИСТИНА). Если же значение *StartAT* уже равно ИСТИНА, когда вход *Run* переходит в состояние ИСТИНА, автонастройка выполняется при запуске ПИД-регулирования.

Рассчитанные значения ПИД-констант применяются сразу после нормального завершения автонастройки.

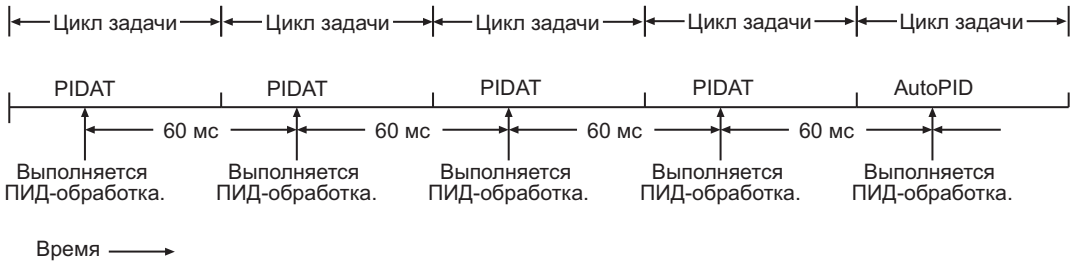
Если значение переменной *StartAT* поменяется на ЛОЖЬ во время автонастройки (т. е. когда *ATBusy* = ИСТИНА), автонастройка будет отменена. После отмены автонастройки ПИД-регулирование запускается снова с предыдущими значениями ПИД-констант.

Время выполнения ПИД-регулирования

ПИД-регулирование выполняется циклически, с заданным периодом выполнения. Обработка в ПИД-регуляторе производится, когда в программе пользователя выполняется команда PIDAT. Однако обработка в ПИД-регуляторе не производится, если после последнего выполнения обработки прошло меньше времени, чем время *SampTime*.
Если же время, прошедшее с момента последнего выполнения, превышает значение *SampTime*, то излишек времени (прошедшее время - *SampTime*) переносится в следующий период. Более подробная информации дана ниже.

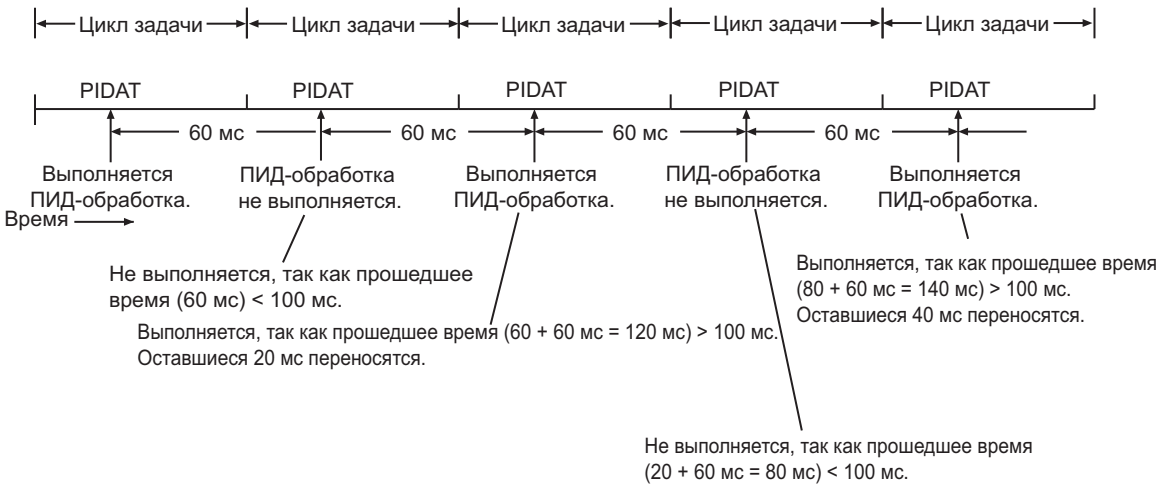
Период выполнения задачи = 60 мс, а ***SampTime*** < 60 мс

Период выполнения задачи больше или равен значению ***SampTime***, поэтому ПИД-обработка выполняется один раз в каждом цикле выполнения задачи.



Период выполнения задачи = 60 мс, а ***SampTime*** =100 мс

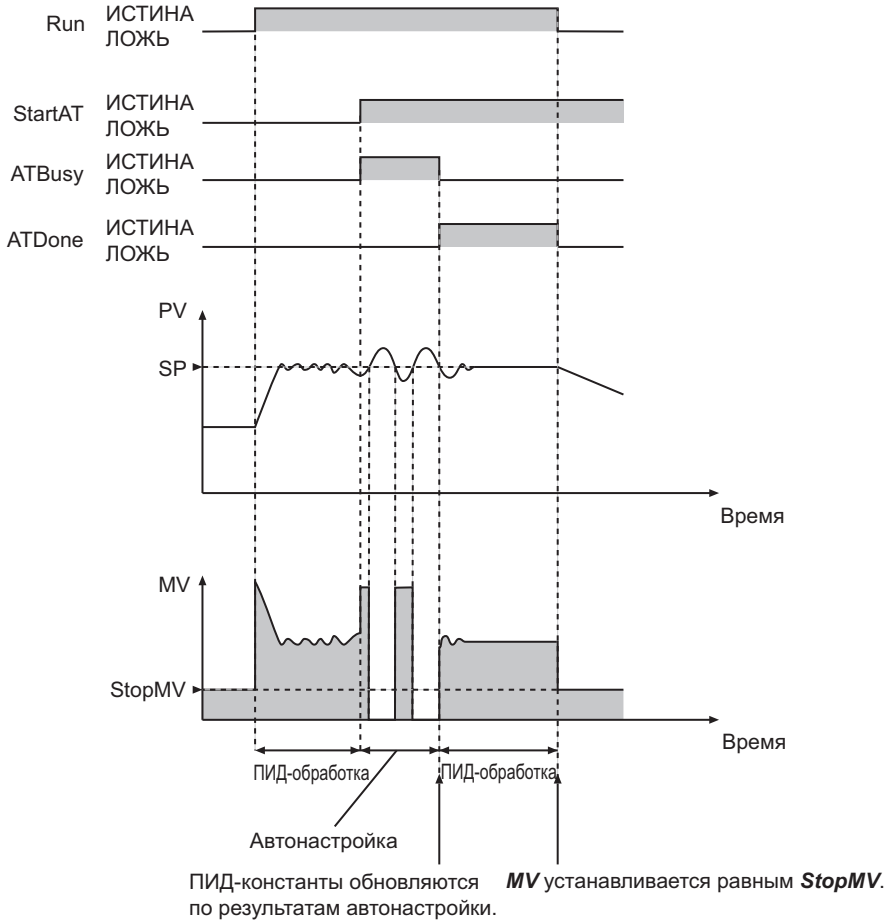
Период выполнения задачи меньше значения ***SampTime***,
поэтому ПИД-обработка выполняется не в каждом цикле выполнения задачи.



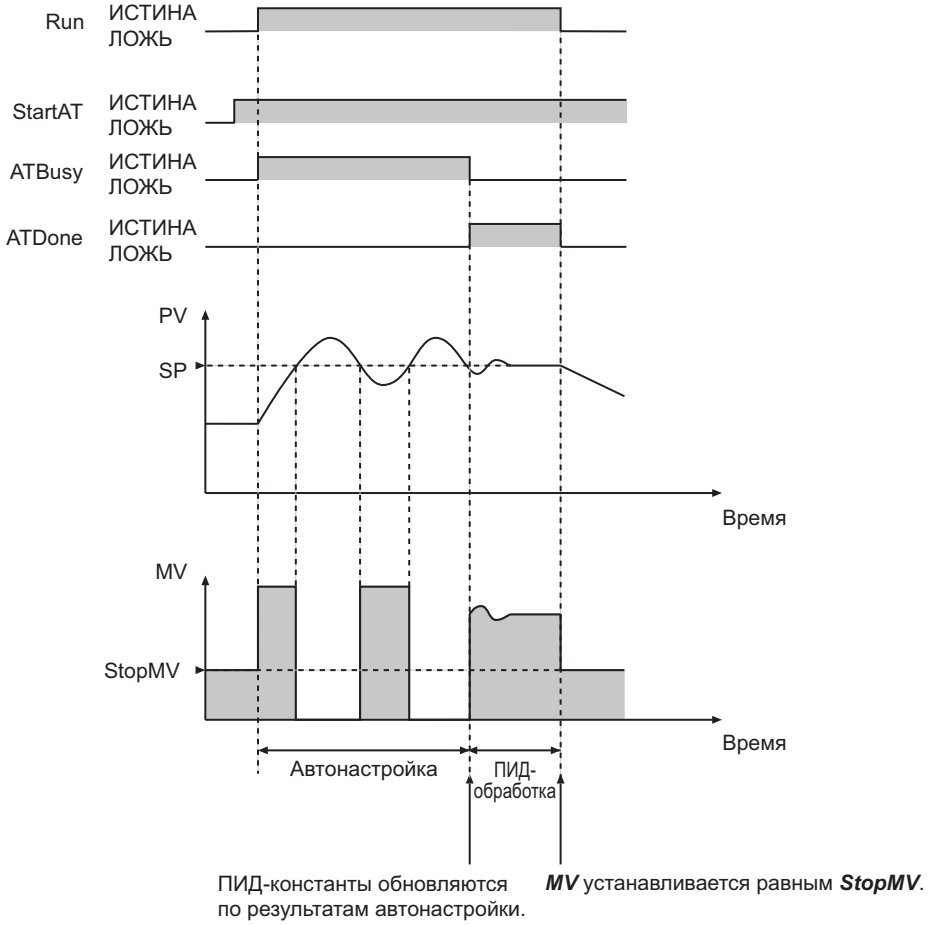
Временные диаграммы

Ниже представлены временные диаграммы изменения переменных команды для разных ситуаций.

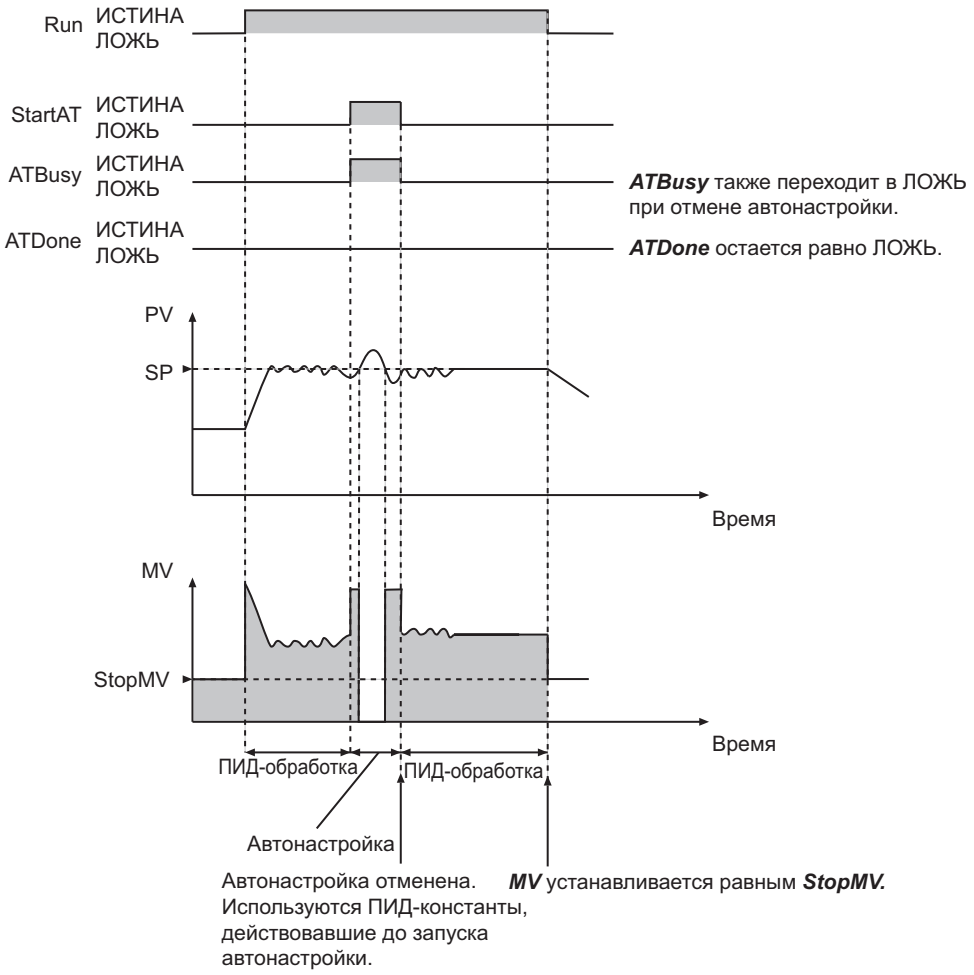
● Выполнение автонастройки во время работы в автоматическом режиме



● Выполнение автонастройки в начале выполнения PIDAT



● Отмена автонастройки

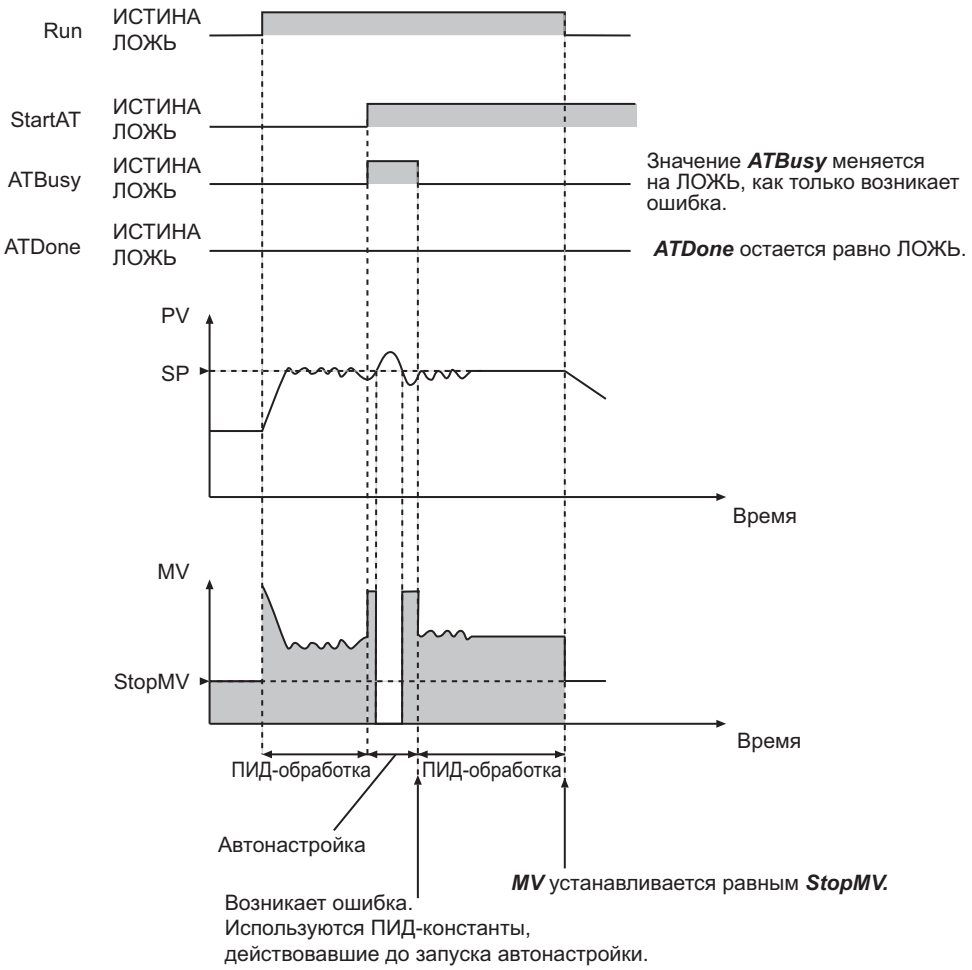


● Возникновение ошибки автонастройки во время автонастройки

В указанных ниже случаях возникает ошибка автонастройки и автонастройка прекращается.

- Если управляющее воздействие равно верхнему предельному значению управляющего воздействия и время достижения нулевого отклонения превышает 19 999 с.
- Если управляющее воздействие равно нижнему предельному значению управляющего воздействия и время выхода отклонения за пределы гистерезиса (*ATHystrs*) превышает 19 999 с.

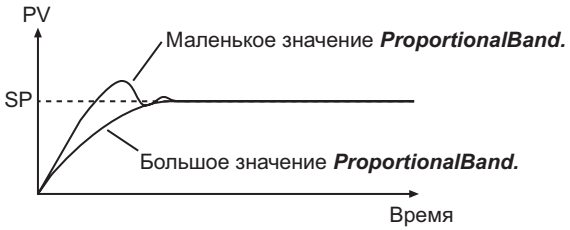
После отмены автонастройки ПИД-регулирование запускается снова с предыдущими значениями ПИД-констант.



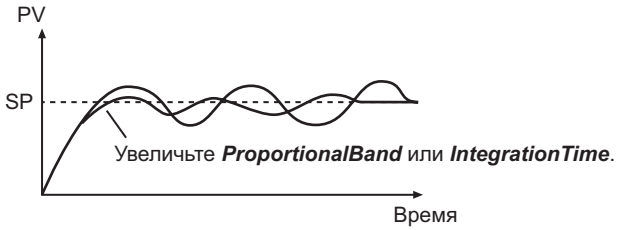
Дополнительная информация

Настройка постоянных ПИД-регулятора

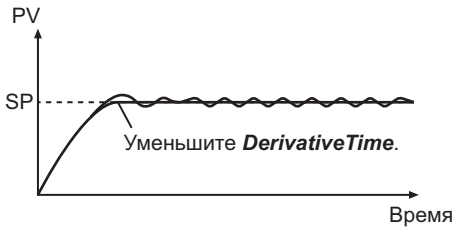
- Если требуется исключить автоколебания, даже если время стабилизации управляемой системы при этом возрастет, следует увеличить значение *ProportionalBand*. Если же автоколебательный процесс не представляет проблемы, но необходимо очень быстро стабилизировать управляемую систему, значение *ProportionalBand* следует уменьшить.



- Для уменьшения длительности автоколебаний следует увеличить *ProportionalBand* или *IntegrationTime*.



- При возникновении автоколебаний большой частоты следует уменьшить значение *DerivativeTime*.



Начальные значения постоянных ПИД-регулятора для регулирования температуры

Ниже приведены начальные значения постоянных ПИД-регулятора, которые рекомендуется использовать в случае применения команды PIDAT для регулирования температуры. Для других переменных следует использовать значения по умолчанию.

Переменные	Начальные значения (справочно)*1
ProportionalBand	10 % от полн. диап.
IntegrationTime	233 с
DerivativeTime	40 с

*1. При выполнении автонастройки следует использовать результаты, полученные при автонастройке.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Значения переменных *PV* и *SP* должны находиться в диапазоне между значениями *RngLowLmt* и *RngUpLmt* включительно. Единицы измерения этих переменных должны быть согласованы в соответствии с таблицей ниже.

Единица	Значения *PV* и *SP*	Значения *RngLowLmt* и *RngUpLmt*
% от полн. диап.	*PV* = (регулируемая величина в физических единицах - MIN)/(MAX - MIN) × 100*1 *SP* = (заданное значение в физических единицах - MIN)/(MAX - MIN) × 100*1	*RngLowLmt* = 0 *RngUpLmt* = 100
Физическая единица	*PV* = регулируемая величина в физических единицах *SV* = заданное значение в физических единицах	*RngLowLmtl* = MIN*1 *RngUpLmt* = MAX*1

*1. МАКС: верхняя граница входного диапазона в физических единицах измерения; МИН: нижняя граница входного диапазона в физических единицах измерения.

- В следующей таблице показано, изменение каких переменных возможно в том или ином рабочем состоянии.

Переменные	Состояние регулятора		
	Выполнение команды остановлено*1	Работа в автоматическом режиме (автонастройка в данный момент не выполняется)*2	Работа в автоматическом режиме (в данный момент выполняется автонастройка)*3
Run	Возможно	Возможно	Возможно
ManCtl	Возможно	Возможно	Возможно
StartAT	Возможно	Возможно	Возможно
PV	Возможно	Возможно	Возможно
SP	Возможно	Возможно	Невозможно
MVLowLmt	Возможно	Возможно	Невозможно
MVUpLmt	Возможно	Возможно	Невозможно
ManResetVal	Возможно	Возможно	Невозможно
MVTrackSw	Возможно	Возможно	Невозможно
MVTrackVal	Возможно	Возможно	Невозможно
StopMV	Возможно	Возможно	Возможно
ErrorMV	Возможно	Возможно	Возможно
Alpha	Возможно	Возможно	Невозможно
ATCalcGain	Возможно	Возможно	Невозможно
ATHystrs	Возможно	Возможно	Невозможно
SampTime	Возможно	Невозможно	Невозможно
RngLowLmt	Возможно	Невозможно	Невозможно
RngUpLmt	Возможно	Невозможно	Невозможно
DirOpr	Возможно	Невозможно	Невозможно
ProportionalBand	Возможно	Возможно	Невозможно
IntegrationTime	Возможно	Возможно	Невозможно
DerivativeTime	Возможно	Возможно	Невозможно
ManMV	Возможно	Возможно	Возможно

*1. *ManCtl* = ИСТИНА, *Run* = ЛОЖЬ, *Error* = ИСТИНА или *MVTrackSw* = ИСТИНА.
*2. *ManCtl* = ЛОЖЬ, *Run* = ИСТИНА, *Error* = ЛОЖЬ, *MVTrackSw* = ЛОЖЬ и *ATBusy* = ЛОЖЬ.
*3. *ManCtl* = ЛОЖЬ, *Run* = ИСТИНА, *Error* = ЛОЖЬ, *MVTrackSw* = ЛОЖЬ и *ATBusy* = ИСТИНА.

- Значение *SampTime* усекается с точностью до 100 нс.
- Если значение *StartAT* поменяется на ИСТИНА, когда значение *ManCtl* = ИСТИНА, автонастройка запустится в следующий раз, когда значение *ManCtl* поменяется на ЛОЖЬ.
- Если значение *ErrorMV* находится за пределами допустимого диапазона (-320...320), значение *MV* будет равно 0 при возникновении ошибки.
- Если значение *ManCtl* поменяется на ИСТИНА во время автонастройки, автонастройка будет отменена.
- Даже если во время автонастройки возникает ошибка, выход *Error* не переходит в состояние ИСТИНА.
- В указанном ниже случае происходит ошибка. Выход *Error* перейдет в состояние ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки. Выходы *ATDone* и *ATBusy* перейдут в состояние ЛОЖЬ. Если входы *ManCtl* и *Run* будут находиться в состоянии ЛОЖЬ, переменная *MV* примет значение *ErrorMV*. Если значение *ErrorMV* будет находиться вне допустимого диапазона, *MV* примет значение 0.

Ошибка	Значение *ErrorID*
Значение входной переменной находится за пределами допустимого диапазона.	16#0400
RngLowLmt больше или равно *RngUpLmt*.	16#0401
MVLowLmt больше или равно *MVUpLmt*.	

- Если работа регулятора должна прекращаться при ошибках, не указанных выше, в программе системы необходимо предусмотреть, чтобы значение переменной *Run* менялось на ЛОЖЬ при возникновении ошибки.
- Если возникает ошибка из-за того, что значение *PV* или *SP* вышло за допустимый диапазон, состояние ошибки длится не менее 5 секунд, даже если значение возвращается в границы допустимого диапазона раньше. Это означает, что выход *Error* будет находиться в состоянии ЛОЖЬ пять секунд.
- Если после сброса ошибки вход *Run* находится в состоянии ИСТИНА, ПИД-регулирование автоматически перезапускается. Если кроме входа *Run* в состоянии ИСТИНА также находится вход *StartAT*, также автоматически перезапускается автонастройка.
- Проверка на наличие ошибок производится в каждом цикле измерения.

Пример программы

В качестве примера будет рассмотрена программа, в которой команда PIDAT применяется для регулирования температуры.
Управляющее воздействие на выходе команды PIDAT преобразуется в сигнал для широтно-импульсного регулирования (ШИР), который подается на нагревательное устройство.
В данном примере для преобразования управляющего воздействия в сигнал ШИР используется команда таймера.
См. *Пример программы* на стр. 2-845 для команды TimeProportionalOut.

Характеристики

В таблице ниже указано используемое оборудование и параметры регулирования температуры.

Устройство/параметр	Характеристики
Тип входа	Термопара типа K
Модуль входов	CJ1W-PH41U (модуль универсальных входов с гальванической развязкой)
Модуль выходов	CJ1W-OD212 (модуль транзисторных выходов)
Заданное значение	90 °C
Период измерения для ПИД-регулирования	100 мс
Период выходного сигнала управления	1 с

Конфигурация и настройки

В таблице ниже приведен используемый параметр модуля входов CJ1W-PH41U.

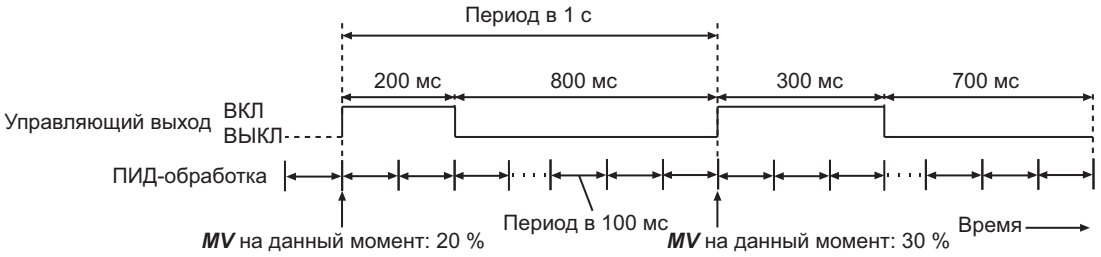
Параметр	Заданное значение
Input1: тип входного сигнала	K(1)

В таблице ниже перечислены используемые параметры карты соответствия входов и выходов (I/O map).

Модуль	Порт ввода-вывода	Описание	Переменная
CJ1W-PH41U	Ch1_AIInPV	Вход 1, регулируемая величина (значение типа INT)	AI1
CJ1W-OD212	Ch1_Out00	Бит 00 выходного слова 1	DO1

Принцип работы

- Переменная *MV* (управляющее воздействие) команды PIDAT используется для управления выходом, который либо включает, либо выключает регулятор температуры.
- Период измерения (*InitSetParams.SampTime*) для команды PIDAT устанавливается равным 100 мс. Период выполнения задачи должен быть меньше (с хорошим запасом), чем 100 мс. Таким образом, значение *MV* обновляется каждые 100 мс.
- Период выходного сигнала управления равен 1 с. В пределах каждого периода управляющий выход включается на определенное время, а затем выключается (широтно-импульсное регулирование). Соотношение между временем включения и временем выключения определяется управляющим воздействием, которое формируется ПИД-регулятором.
  Например, если полученное значение *MV* = 20 %, выход управления регулятором температуры будет включен в течение 200 мс и будет выключен следующие 800 мс.
  Это повторяется с периодом в 1 с.



- Если последнее полученное значение *MV* меньше, чем значение *MV* при определении последних значений времени включения/выключения управляющего выхода, значения времени включения/выключения не изменяются.
  А если последнее полученное значение *MV* больше, чем оно было при определении последних значений времени включения/выключения, значения времени включения/выключения сразу же изменяются с учетом последнего полученного значения *MV*. Пусть, например, последние значения времени включения/выключения были определены, когда значение *MV* было равно 20 % (200 мс — ВКЛ и 800 мс — ВЫКЛ).
  Если по истечении 100 мс новое значение *MV* = 30 %, значения времени включения/выключения выхода немедленно изменяются: время включения становится равно 300 мс, а время выключения — 700 мс.

- Если выполняется автонастройка и значение *MV* становится равно 100 %, выход немедленно включается независимо от периода управления.

Определения глобальных переменных

● Глобальные переменные

Переменная	Тип данных	Параметр «AT» *1	Комментарий
AI1	INT	IOBus://rack#0/slot#0/Ch1_AIInPV	Вход 1, регулируемая величина (значение типа INT)
DO1	BOOL	IOBus://rack#0/slot#1/Ch1_Out/Ch1_Out00	Бит 00 выходного слова 1

*1. В таблице приведены переменные для модуля входов CJ1W-PH41U, установленного в слот №0 стойки №0, и для модуля выходов CJ1W-OD212, установленного в слот №1 той же стойки.

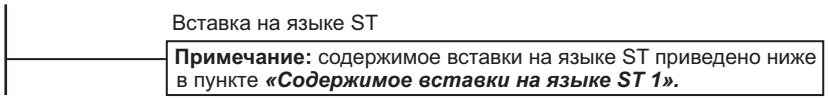
Примечание Глобальные переменные для порта каждого модуля генерируются автоматически на основе настроенного соответствия входов-выходов.

Программа на языке LD

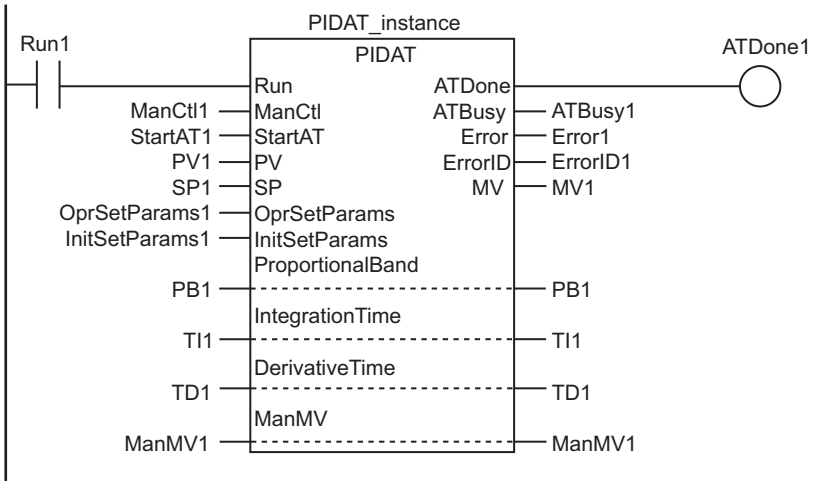
Переменная	Тип данных	Начальное значение	Сохранение	Комментарий
Run1	BOOL	ЛОЖЬ	☐	Условие выполнения
ManCtl1	BOOL	ЛОЖЬ	☐	Ручное/автоматическое управление
StartAT1	BOOL	ЛОЖЬ	☐	Условие выполнения автонастройки
PV1	REAL	0,0	☐	Регулируемая величина
SP1	REAL	90	☐	Заданное значение
OprSetParams1	_sOPR_SET_PARAMS	(MVLowLmt:=0.0, MVUpLmt:=100.0, ManResetVal:=0.0, MVTrackSw:=FALSE, MVTrackVal:=0.0, StopMV:=0.0, ErrorMV:=0.0, Alpha:=0.65, ATCalcGain:=1.0, ATHystrs:=0.2)	☐	Рабочие параметры
InitSetParams1	_sINIT_SET_PARAMS	(SampTime:=T#100 ms, RngLowLmt:=0.0, RngUpLmt:=1000.0, DirOpr:=FALSE)	☐	Начальные параметры
PB1	REAL	10	☑	Зона пропорциональности
TI1	TIME	T#0 c	☑	Постоянная времени интегрирования
TD1	TIME	T#0 c	☑	Постоянная времени дифференцирования
ManMV1	REAL	0,0	☐	Ручное управляющее воздействие
ATDone1	BOOL	ЛОЖЬ	☐	Нормальное завершение автонастройки

Переменная	Тип данных	Начальное значение	Сохранение	Комментарий
ATBusy1	BOOL	ЛОЖЬ	☐	Текущее выполнение автонастройки
Error1	BOOL	ЛОЖЬ	☐	Ошибка
ErrorID1	WORD	16#0	☐	Идентификатор ошибки
MV1	REAL	0,0	☐	Управляющее воздействие
PulseOnTime	TIME	T#0 с	☐	Время включения управляющего выхода
PulseCycTime	TIME	T#1 с	☐	Период регулирования
ResetPulse	BOOL	ЛОЖЬ	☐	Сброс таймера
PIDAT_instance	PIDAT		☐	
TOF_instance	TOF		☐	
TON_instance	TON		☐	

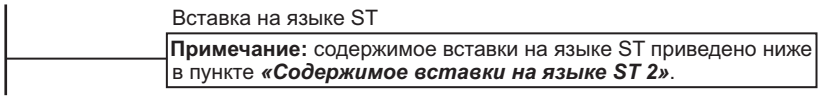
Получение значения регулируемой величины.



Выполнение команды PIDAT.



Выход широтно-импульсного регулирования



● Содержимое вставки на языке ST 1

```
PV1:=INT_TO_REAL(AI1)/REAL#10.0;  // Преобразование AI1 (рег. велич.) в вещественно
е число.
// CJ1W-PH41U выдает значение, которое в 10 раз больше фактического, поэтому его ну
жно поделить на 10.0.
```

● Содержимое вставки на языке ST 2

```
// Расчет времени включения управляющего выхода.
PulseOnTime:=MULTIME(PulseCycTime, MV1/REAL#100.0);
// Переключение между ВКЛ и ВЫКЛ с помощью команды TOF.
TOF_instance(In:=BOOL#FALSE, PT:=PulseOnTime, Q=>DO1);
// Измерение времени сброса таймера с помощью команды TON.
TON_instance(In:=BOOL#TRUE, PT:=PulseCycTime, Q=>ResetPulse);
// Сброс таймера.
IF (ResetPulse=BOOL#TRUE) THEN
  TOF_instance(In:=BOOL#TRUE);
  TON_instance(In:=BOOL#FALSE);
END_IF;
// Если MV1 = 100% для автонастройки.
IF ( (ATBusy1=BOOL#TRUE) & (MV1=REAL#100.0) ) THEN
  DO1:=BOOL#TRUE;  // Немедленное включение выхода.
END_IF;
```

Программа на языке ST

Переменная	Тип данных	Начальное значение	Сохранение	Комментарий
Run1	BOOL	ЛОЖЬ	☐	Условие выполнения
ManCtl1	BOOL	ЛОЖЬ	☐	Ручное/автоматическое управление
StartAT1	BOOL	ЛОЖЬ	☐	Условие выполнения автонастройки
PV1	REAL	0,0	☐	Регулируемая величина
SP1	REAL	90	☐	Заданное значение
OprSetParams1	_sOPR_SET_PARAMS	(MVLowLmt:=0.0, MVUpLmt:=100.0, ManResetVal:=0.0, MVTrackSw:=FALSE, MVTrackVal:=0.0, StopMV:=0.0, ErrorMV:=0.0, Alpha:=0.65, ATCalcGain:=1.0, ATHystrs:=0.2)	☐	Рабочие параметры
InitSetParams1	_sINIT_SET_PARAMS	(SampTime:=T#100 ms, RngLowLmt:=0.0, RngUpLmt:=1000.0, DirOpr:=FALSE)	☐	Начальные параметры
PB1	REAL	10	☑	Зона пропорциональности
TI1	TIME	T#0 c	☑	Постоянная времени интегрирования

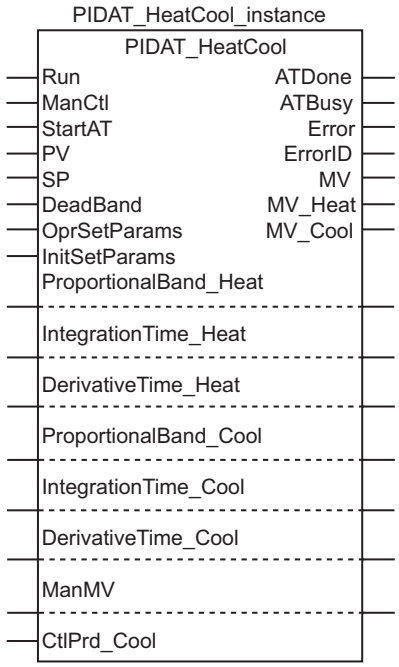
Переменная	Тип данных	Начальное значение	Сохранение	Комментарий
TD1	TIME	T#0 c	☑	Постоянная времени дифференцирования
ManMV1	REAL	0,0	☐	Ручное управляющее воздействие
ATDone1	BOOL	ЛОЖЬ	☐	Нормальное завершение автонастройки
ATBusy1	BOOL	ЛОЖЬ	☐	Текущее выполнение автонастройки
Error1	BOOL	ЛОЖЬ	☐	Ошибка
ErrorID1	WORD	16#0	☐	Идентификатор ошибки
MV1	REAL	0,0	☐	Управляющее воздействие
PulseOnTime	TIME	T#0 c	☐	Время включения управляющего выхода
PulseCycTime	TIME	T#1 c	☐	Период регулирования
ResetPulse	BOOL	ЛОЖЬ	☐	Сброс таймера
PIDAT_instance	PIDAT		☐	
TOF_instance	TOF		☐	
TON_instance	TON		☐	

```
// Преобразование AI1 (рег. велич.) в вещественное число.
// CJ1W-PH41U выдает значение, которое в 10 раз больше фактического, поэтому его ну
жно поделить на 10.0.
PV1:=INT_TO_REAL(AI1)/REAL#10.0;

// Выполнение команды PIDAT.
PIDAT_instance(
  Run :=Run1,
  ManCtl :=ManCtl1,
  StartAT :=StartAT1,
  PV :=PV1,
  SP :=SP1,
  OprSetParams :=OprSetParams1,
  InitSetParams :=InitSetParams1,
  ProportionalBand:=PB1,
  IntegrationTime :=TI1,
  DerivativeTime :=TD1,
  ManMV :=ManMV1,
  ATDone =>ATDone1,
```

```
  ATBusy =>ATBusy1,
  Error =>Error1,
  ErrorID =>ErrorID1,
  MV =>MV1);

// Выход ШИР
// Расчет времени включения управляющего выхода.
PulseOnTime:=MULTIME(PulseCycTime, MV1/REAL#100.0);
// Переключение между ВКЛ и ВЫКЛ с помощью команды TOF.
TOF_instance(In:=BOOL#FALSE, PT:=PulseOnTime, Q=>DO1);
// Переключение между ВКЛ и ВЫКЛ с помощью команды TOF.
TON_instance(In:=BOOL#TRUE, PT:=PulseCycTime, Q=>ResetPulse);
// Сброс таймера.
IF (ResetPulse=BOOL#TRUE) THEN
  TOF_instance(In:=BOOL#TRUE);
  TON_instance(In:=BOOL#FALSE);
END_IF;
// Если MV1 = 100% для автонастройки.
IF ( (ATBusy1=BOOL#TRUE) & (MV1=REAL#100.0) ) THEN
  DO1:=BOOL#TRUE;  // Немедленное включение выхода.
END_IF;
```

PIDAT_HeatCool

Команда PIDAT_HeatCool реализует ПИД-регулятор нагрева/охлаждения с автонастройкой (2-ПИД-регулятор с фильтром уставки).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
PIDAT_HeatCool	ПИД-регулятор нагрева/охлаждения с автонастройкой	FB		PIDAT_HeatCool_instance(Run, ManCtl, StartAT, PV, SP, DeadBand, OprSetParams, InitSetParams, ProportionalBand_Heat, IntegrationTime_Heat, DerivativeTime_Heat, ProportionalBand_Cool, IntegrationTime_Cool, DerivativeTime_Cool, ManMV, CtlPrd_Cool, ATDone, ATBusy, Error, ErrorID, MV, MV_Heat, MV_Cool);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Run	Условие выполнения	Вход	ИСТИНА: выполнить ЛОЖЬ: остановить	Зависит от типа данных.	---	ЛОЖЬ
ManCtl	Ручное/автоматическое управление		ИСТИНА: работа в режиме ручного управления ЛОЖЬ: работа в автоматическом режиме			
StartAT	Условие выполнения автонастройки		ИСТИНА: выполнить ЛОЖЬ: отменить			
PV	Регулируемая величина		Регулируемая величина	*1		0
SP	Заданное значение		Заданное значение			
DeadBand	Зона нечувствительности		Настройка зоны нечувствительности/ зоны перекрытия	-320,0...320,0	%	
OprSetParams	Рабочие параметры		Параметры, задаваемые во время работы	---	---	---
InitSetParams	Начальные параметры		Начальные параметры			
CtlPrd_Cool	Период регулирования охлаждения		Период регулирования при использовании *MV_Cool* для широтно-импульсного регулирования.	T#0.1 с ... T#100 с	---	T#20 с

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Proportional Band_Heat	Зона пропорциональности для регулирования нагрева	Вход-выход	Зона пропорциональности для регулирования нагрева	0,01...1000,00	% от полн. диап.	---
IntegrationTime_Heat	Постоянная времени интегрирования для регулирования нагрева		Постоянная времени интегрирования для регулирования нагрева Чем больше это значение, тем меньше управляющее воздействие по интегралу. При значении 0 интегральное звено отключено.	T#0.0000 с ... T#10000.0000 с*2	с	
Derivative Time_Heat	Постоянная времени дифференцирования для регулирования нагрева		Постоянная времени дифференцирования для регулирования нагрева Чем больше это значение, тем больше управляющее воздействие по производной. При значении 0 дифференциальное звено отключено.	T#0.0000 с ... T#10000.0000 с*2		
Proportional Band_Cool	Зона пропорциональности для регулирования охлаждения		Зона пропорциональности для регулирования охлаждения	0,01...1000,00	% от полн. диап.	
IntegrationTime_Cool	Постоянная времени интегрирования для регулирования охлаждения		Постоянная времени интегрирования для регулирования охлаждения Чем больше это значение, тем меньше управляющее воздействие по интегралу. При значении 0 интегральное звено отключено.	T#0.0000 с ... T#10000.0000 с*2	с	
Derivative Time_Cool	Постоянная времени дифференцирования для регулирования охлаждения		Постоянная времени дифференцирования для регулирования охлаждения Чем больше это значение, тем больше управляющее воздействие по производной. При значении 0 дифференциальное звено отключено.	T#0.0000 с ... T#10000.0000 с*2		
ManMV	Ручное управляющее воздействие		Ручное управляющее воздействие	-320...320	%	

	Значение	Вход/выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
ATDone	Нормальное завершение автонастройки	Выход	ИСТИНА: нормальное завершение ЛОЖЬ:*3	Зависит от типа данных.	---	---
ATBusy	Выполнение автонастройки		ИСТИНА: выполняется автоматическая настройка ЛОЖЬ: автонастройка не выполняется			
MV	Управляющее воздействие		Управляющее воздействие		%	
MV_Heat	Управляющее воздействие для регулирования нагрева		Управляющее воздействие для регулирования нагрева	0...320		
MV_Cool	Управляющее воздействие для регулирования охлаждения		Управляющее воздействие для регулирования охлаждения	0...320		

*1. От значения *InitSetParams.RngLowLmt* (нижняя граница входного диапазона) до значения *InitSetParams.RngUpLmt* (верхняя граница входного диапазона).
*2. Значение усекается до четырех знаков после запятой.
*3. Значение ЛОЖЬ означает, что команда завершена с ошибкой, что в данный момент выполняется ПИД-регулирование без автонастройки или что в данный момент ПИД-регулирование не выполняется.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Run	OK																			
ManCtl	OK																			
StartAT	OK																			
PV														OK						
SP														OK						
DeadBand														OK						
OprSetParams	Подробные сведения о структуре _sOPR_SET_PARAMS см. в разделе *Описание структурного типа* на стр. 2-802.																			
InitSetParams	Подробные сведения о структуре _sINIT_SET_PARAMS см. в разделе *Описание структурного типа* на стр. 2-802.																			
CtlPrd_Cool																OK				
Proportional Band_Heat														OK						
IntegrationTime_Heat																OK				
Derivative Time_Heat																OK				
Proportional Band_Cool														OK						
IntegrationTime_Cool																OK				

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Derivative Time_Cool																ОК				
ManMV														ОК						
ATDone	ОК																			
ATBusy	ОК																			
MV														ОК						
MV_Heat														ОК						
MV_Cool														ОК						

Функция

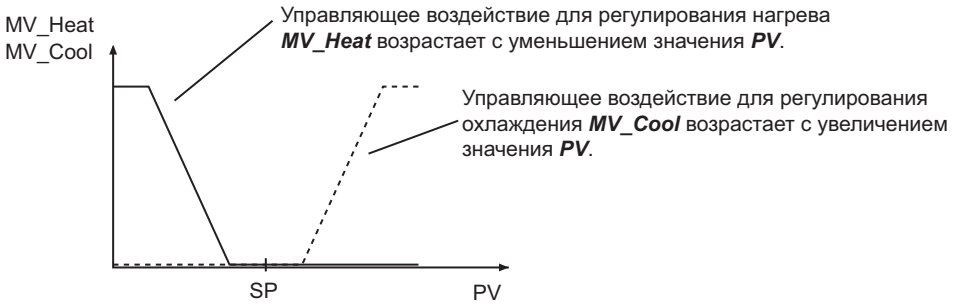
Команда PIDAT_HeatCool осуществляет пропорционально-интегрально-дифференциальное регулирование (ПИД-регулирование) нагрева/охлаждения и формирует управляющее воздействие для регулятора температуры или другого устройства.
ПИД-регулирование нагрева/охлаждения запускается, когда значение переменной *Run* (условие выполнения) меняется на ИСТИНА. Пока значение *Run* = ИСТИНА, циклически повторяется следующая рабочая последовательность: считывается текущее значение регулируемой величины *PV*, выполняются расчеты ПИД-регулятора нагрева/охлаждения («ПИД-обработка») и на выходы *MV_Heat* и *MV_Cool* подаются рассчитанные управляющие воздействия для нагрева и охлаждения соответственно.
Когда значение переменной *Run* меняется на ЛОЖЬ, ПИД-регулирование нагрева/охлаждения останавливается.

Поддерживается функция автоматической настройки для автоматического определения оптимальных значений постоянных ПИД-регулятора для регулирования нагрева и регулирования охлаждения.
Автоматическая настройка постоянных ПИД-регулятора для регулирования нагрева и регулирования охлаждения производится, когда значение переменной *StartAT* (условие выполнения автонастройки) меняется на ИСТИНА.

Различия между командами PIDAT_HeatCool и PIDAT

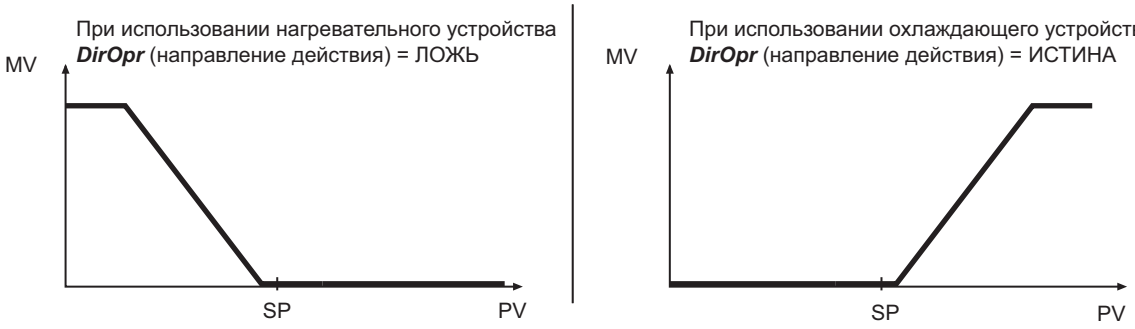
● Команда PIDAT_HeatCool

Команда PIDAT_HeatCool рассчитана на одновременное использование нагревательного и охлаждающего устройств для регулирования температуры. Поэтому она выдает два разных управляющих воздействия: управляющее воздействие для регулирования нагрева (*MV_Heat*) и управляющее воздействие для регулирования охлаждения (*MV_Cool*). Функция автонастройки определяет оптимальные ПИД-константы для регулирования нагрева и оптимальные ПИД-константы для регулирования охлаждения.



● Команда PIDAT

Команда PIDAT рассчитана на использование только одного устройства для регулирования температуры: нагревательного устройства или охлаждающего устройства. Поэтому она имеет только один выход управляющего воздействия (*MV*). В ней также предусмотрен параметр *DirOpr* (направление действия), который определяет, будет ли управляющее воздействие подаваться на нагревательное устройство или на охлаждающее устройство. В команде PIDAT_HeatCool параметр *DirOpr* не предусмотрен.

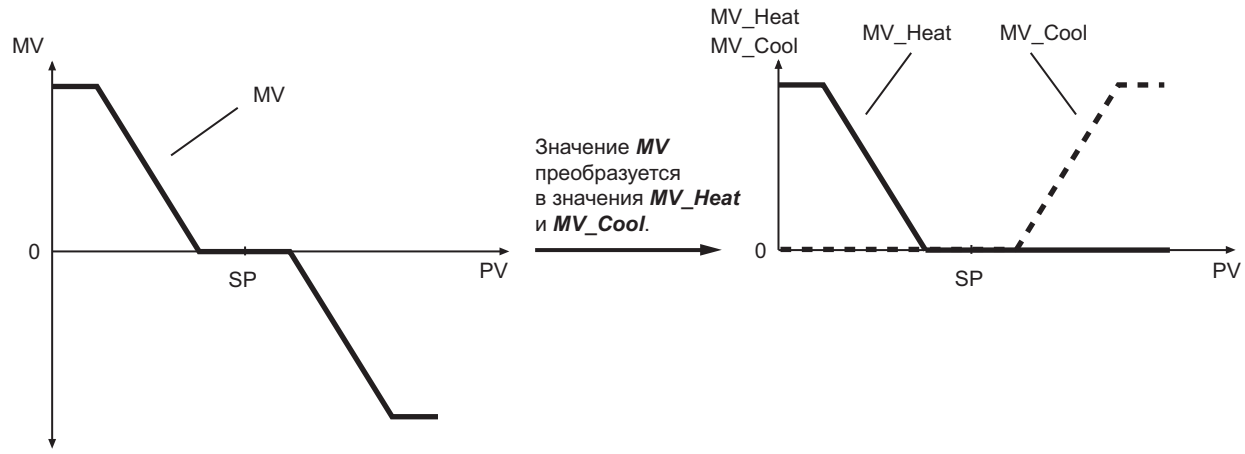


Управляющее воздействие *MV*, управляющее воздействие для регулирования нагрева *MV_Heat* и управляющее воздействие для регулирования охлаждения *MV_Cool*

Переменная *MV* содержит управляющее воздействие для регулирования температуры для случая, когда в системе есть только нагревательное устройство или только охлаждающее устройство (см. также сравнение команд ПИД-регулирования выше).
Команда PIDAT_HeatCool вычисляет значение *MV* таким же образом, как и команда PIDAT. Управляющее воздействие *MV* преобразуется в управляющие воздействия для нагревательного и охлаждающего устройств (*MV_Heat* и *MV_Cool* соответственно).

Принцип преобразования значения *MV* в значения *MV_Heat* и *MV_Cool* представлен в графическом виде на рисунке ниже.

Значение *MV_Cool* — это абсолютное значение *MV*, когда последнее отрицательно.



На рисунке выше показан только общий принцип преобразования. Фактические значения *MV_Heat* и *MV_Cool* несколько отличаются от абсолютного значения *MV*.

Значения *MV_Heat* и *MV_Cool* рассчитываются на основе значения *MV* по специальным формулам.

Описание структурного типа

Для переменной **OprSetParams** (рабочие параметры) используется структурный тип данных _sOPR_SET_PARAMS.
Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
OprSetParams	Рабочие параметры	Параметры, задаваемые во время работы.	_sOPR_SET_PARAMS	---	---	---

Имя		Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
	MVLowLmt	Нижний предел управляющего воздействия	Нижнее предельное значение *MV_Heat* и *MV_Cool*	REAL	-320...320*1	%	-100
	MVUpLmt	Верхний предел управляющего воздействия	Верхнее предельное значение *MV_Heat* и *MV_Cool*	REAL			100
	ManResetVal	Значение ручного сброса	Не используется.	REAL	-320...320		0
	MVTrackSw	Переключатель отслеживания управляющего воздействия (MV)	Переключатель отслеживания управляющего воздействия (MV) ИСТИНА: ВКЛ ЛОЖЬ: ВЫКЛ	BOOL	Зависит от типа данных.	---	ЛОЖЬ
	MVTrackVal	Отслеживаемое значение управляющего воздействия (MV)	Значение, устанавливаемое в переменной *MV* во время отслеживания управляющего воздействия.	REAL	-320...320	%	0
	StopMV	Управляющее воздействие при остановке	Значение, устанавливаемое в переменной *MV*, когда выполнение команды остановлено.	REAL			
	ErrorMV	Управляющее воздействие при ошибке	Значение, устанавливаемое в переменной *MV*, когда возникает ошибка.	REAL			
	Alpha	Параметр α 2-ПИД регулятора	Коэффициент α фильтра уставки. Если это значение равно 0, фильтр уставки отключен.	REAL	0,00...1,00	---	0,65
	ATCalcGain	Коэффициент влияния автонастройки	Коэффициент регулировки по результатам автонастройки. При более высоких значениях более высоким приоритетом обладает стабильность. При меньших значениях более приоритетной является скорость реакции.	REAL	0,1...10,0		0,8
	ATHystrs	Гистерезис автонастройки	Гистерезис предельного цикла.	REAL	0,01...10,0	% от полн. диап.	0,05

*1. Значение *MVLowLmt* должно быть меньше значения *MVUpLmt*.

Для переменной **InitSetParams** (начальные параметры) используется структурный тип данных _sINIT_SET_PARAMS. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
InitSetParams	Начальные параметры	Начальные параметры.	_sINIT_SET_PARAMS	---	---	---
SampTime	Период измерения	Период выполнения ПИД-обработки.	TIME	T#0.0001 с ... #100.0000 с	с	T#0.05 с
RngLowLmt	Нижняя граница входного диапазона	Нижнее предельное значение переменных *PV* и *SP*.	REAL	-32000...32000*1	---	0
RngUpLmt	Верхняя граница входного диапазона	Верхнее предельное значение переменных *PV* и *SP*.	REAL			100
DirOpr	Направление действия	Не используется.	BOOL	Зависит от типа данных.		ЛОЖЬ

*1. Значение *RngLowLmt* должно быть меньше значения *RngUpLmt*.

Назначение переменных

Ниже поясняется назначение переменных, которые используются в данной команде.

● Run (условие выполнения)

Эта переменная играет роль условия выполнения команды.
ПИД-регулирование нагрева/охлаждения производится, когда она содержит значение ИСТИНА. Когда ее значение меняется на ЛОЖЬ, ПИД-регулирование нагрева/охлаждения останавливается.

● ManCtl (ручное/автоматическое управление)

Эта команда может выполняться в одном из двух режимов: в режиме ручного управления или в автоматическом режиме.
Значение переменной *ManCtl* определяет используемый режим.

Значение *ManCtl*	Режим работы	Значение *MV*
ИСТИНА	Ручное управление	Значение *ManMV* ПИД-регулирование нагрева/охлаждения не производится.
ЛОЖЬ	Автоматическое управление	Значение, рассчитанное ПИД-регулятором нагрева/охлаждения

● StartAT (условие выполнения автонастройки)

Эта переменная играет роль условия выполнения автоматической настройки постоянных ПИД-регулятора.
Если переменная *StartAT* уже содержит значение ИСТИНА в момент перехода переменной *Run* в состояние ИСТИНА, автоматическая настройка констант выполняется при запуске ПИД-регулятора.

Если же значение *StartAT* меняется на ИСТИНА во время ПИД-регулирования нагрева/охлаждения (т. е. когда значение *Run* = ИСТИНА), автоматическая настройка выполняется во время ПИД-регулирования нагрева/охлаждения.
И в том и в другом случае автонастройка отменяется, если значение *StartAT* меняется на ЛОЖЬ во время автонастройки.
Дополнительные сведения о функции автоматической настройки см. в разделе *Автоматическая настройка* на стр. 2-815.

● PV (регулируемая величина)

Это регулируемый технологический параметр управляемой системы.

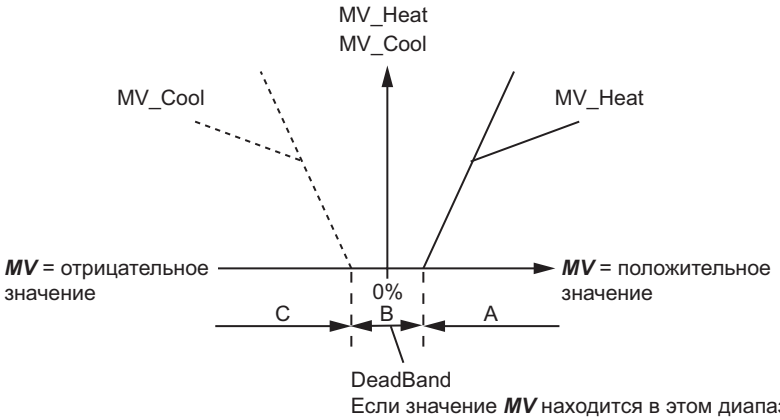
● SP (уставка)

Это заданное значение технологического параметра управляемой системы.

● DeadBand (зона нечувствительности)

Переменная *DeadBand* определяет, как значение *MV* преобразуется в значения *MV_Heat* и *MV_Cool*.
Параметр *DeadBand* задает диапазон значений *MV* с центром в точке *MV* = 0, в пределах которого операции нагрева и охлаждения не выполняются.
В следующей таблице и на рисунке под таблицей поясняется взаимосвязь между значением *MV* и значениями *MV_Heat* и *MV_Cool*.

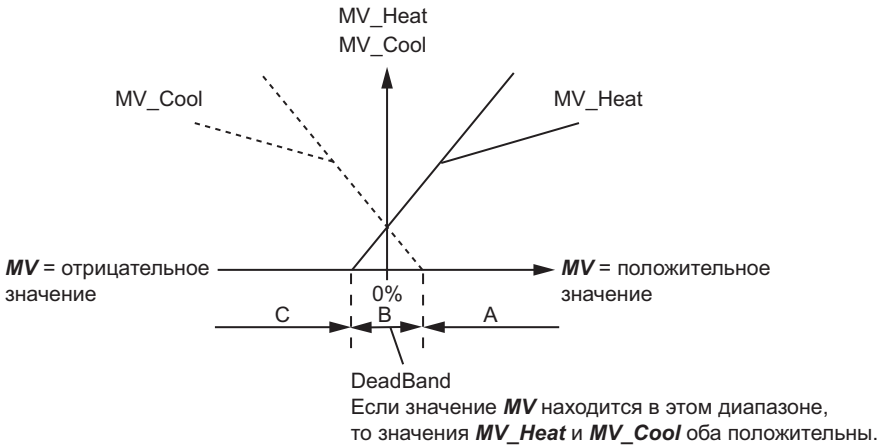
Значение *MV*	Значение *MV_Heat*	Значение *MV_Cool*
Больше, чем зона нечувствительности (область A)	Положительное. Увеличивается при увеличении значения *MV*.	0
В пределах зоны нечувствительности (область B)	0	0
Меньше, чем зона нечувствительности (область C)	0	Положительное. Увеличивается при уменьшении значения *MV*.



Если значение ***MV*** находится в этом диапазоне, то значения ***MV_Heat*** и ***MV_Cool*** равны 0 %.

Для переменной *DeadBand* также можно задать отрицательное значение.
Если значение *DeadBand* отрицательно, а значение *MV* находится в пределах зоны нечувствительности, выполняется одновременно регулирование нагрева и регулирование охлаждения.
В следующей таблице и на рисунке под таблицей поясняется взаимосвязь между значением *MV* и значениями *MV_Heat* и *MV_Cool*, когда значение *DeadBand* отрицательно.

Значение *MV*	Значение *MV_Heat*	Значение *MV_Cool*
Больше, чем зона нечувствительности (область A)	Положительное. Увеличивается при увеличении значения *MV*.	0
В пределах зоны нечувствительности (область B)	Положительное. Увеличивается при увеличении значения *MV*.	Положительное. Увеличивается при уменьшении значения *MV*.
Меньше, чем зона нечувствительности (область C)	0	Положительное. Увеличивается при уменьшении значения *MV*.



● MVLowLmt (нижний предел MV) и MVUpLmt (верхний предел MV)

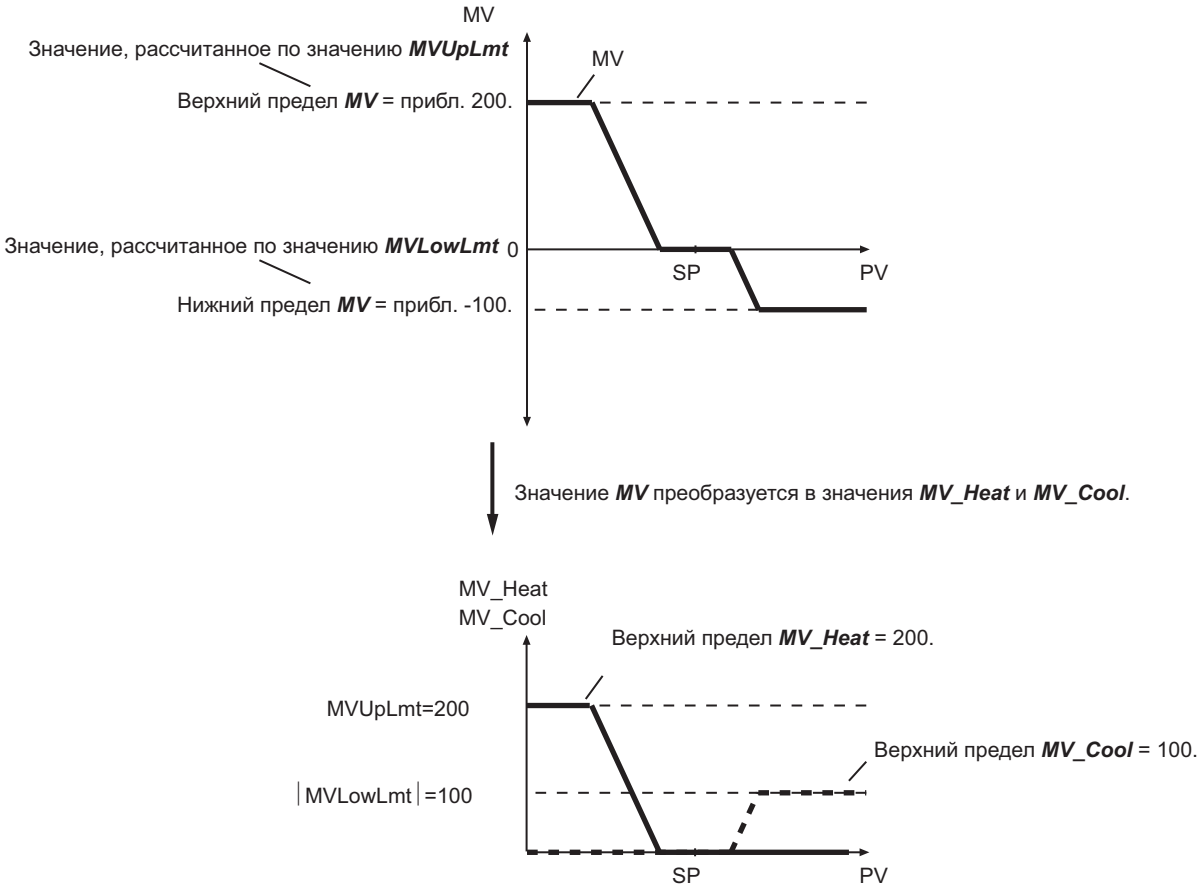
С помощью этих переменных можно ограничить диапазон значений переменных *MV_Heat* и *MV_Cool*.
Параметры *MVLowLmt* и *MVUpLmt* задают соответственно верхнее и нижнее предельные значения для переменных *MV_Heat* и *MV_Cool*.

Для определения значений *MV_Heat* и *MV_Cool* используется приведенная ниже процедура.

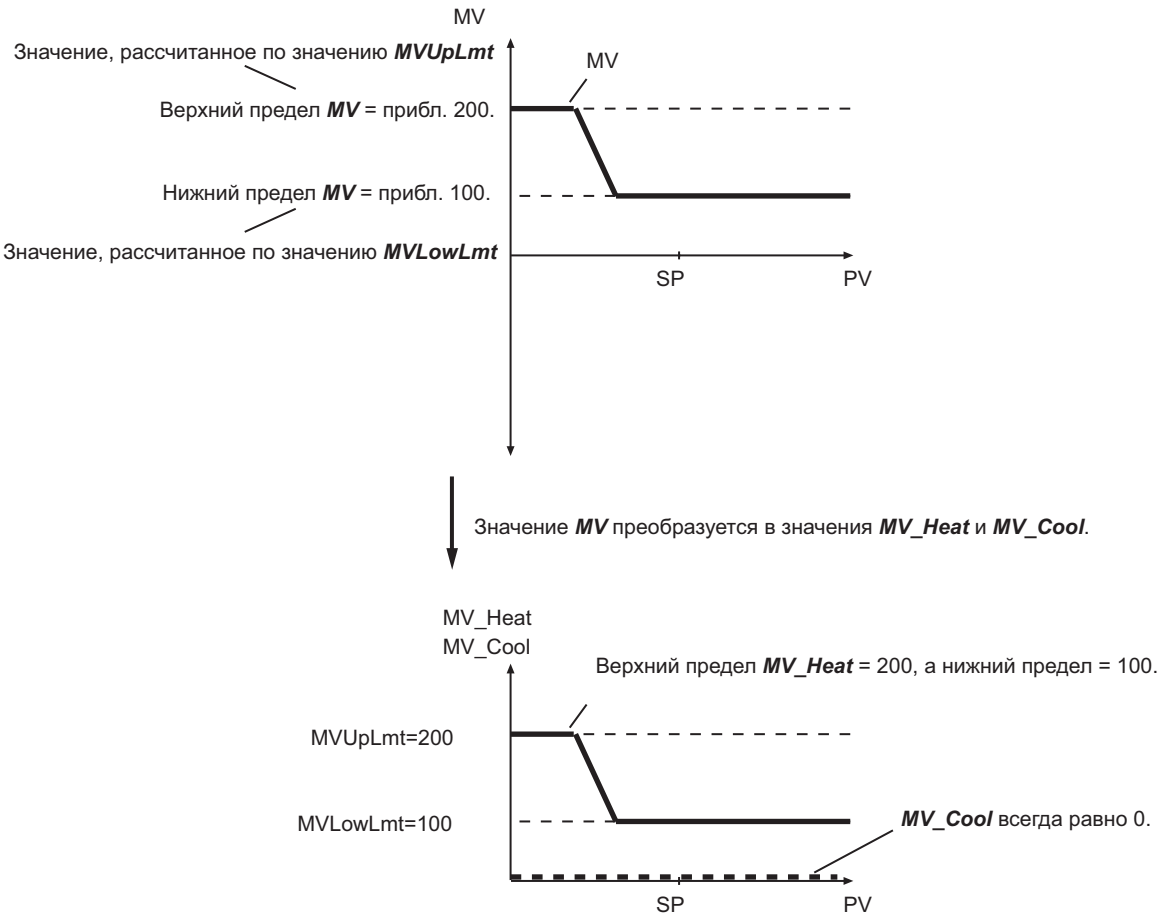
1 В ПИД-регуляторе нагрева/охлаждения вычисляется значение *MV*. На основе значений *MVLowLmt* и *MVUpLmt* по специальным формулам рассчитываются верхнее и нижнее предельные значения *MV*.

2 Значения переменных *MV_Heat* и *MV_Cool* определяются в результате преобразования значения *MV*.

На рисунке ниже показана взаимосвязь между значениями *MV*, *MV_Heat* и *MV_Cool* для случая, когда *MVLowLmt* = -100, а *MVUpLmt* = 200.
Рассчитанный верхний предел *MV_Heat* = 200, а рассчитанный нижний предел = 0. Рассчитанный верхний предел *MV_Cool* = 100, а рассчитанный нижний предел = 0. Другими словами, верхний предел *MV_Heat* совпадает со значением *MVUpLmt*, а верхний предел *MV_Cool* равен абсолютному значению *MVLowLmt*.



На рисунке ниже показана взаимосвязь между значениями *MV*, *MV_Heat* и *MV_Cool* для случая, когда *MVLowLmt* = 100, а *MVUpLmt* = 200.
Рассчитанный верхний предел *MV_Heat* = 200, а рассчитанный нижний предел = 100. Значение *MV_Cool* всегда равно 0. Другими словами, верхний и нижний пределы *MV_Heat* совпадают со значениями *MVUpLmt* и *MVLowLmt* соответственно.



Как видно из рисунка выше, верхний и нижний пределы *MV_Heat* и *MV_Cool* изменяются в зависимости от знака значений *MVLowLmt* и *MVUpLmt*. См таблицу ниже.

Значение *MVLowLmt*	Значение *MVUpLmt*	MV_Heat		MV_Cool	
		Нижний предел	Верхний предел	Нижний предел	Верхний предел
Положительное	Положительное	MVLowLmt	MVUpLmt	0	0
Отрицательное	Положительное	0	MVUpLmt	0	Абсолютное значение *MVLowLmt*
Отрицательное	Отрицательное	0	0	Абсолютное значение *MVUpLmt*	Абсолютное значение *MVLowLmt*

Обязательно задавайте значения *MVLowLmt* и *MVUpLmt* так, чтобы *MVLowLmt* было меньше *MVUpLmt*.
Если переменной *MV* присвоено значение *StopMV*, *ErrorMV* или *ManMV*, ограничение не применяется.

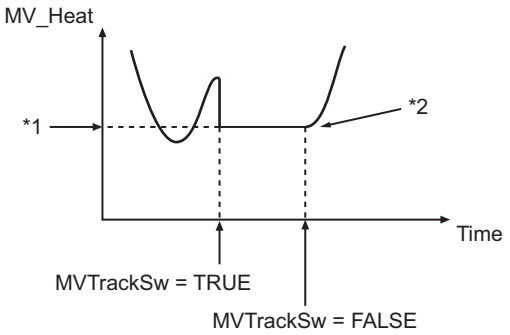
Значения *MVLowLmt* и *MVUpLmt* можно изменять, даже если команда в данный момент не находится в состоянии «автоматическая настройка при работе в автоматическом режиме».
Однако если *MVLowLmt* или *MVUpLmt* будут изменены в сторону расширения диапазона во время работы команды, будет выдано такое же значение *MV_Heat* или *MV_Cool*, как в последнем цикле измерения, а затем оно плавно («безударно») изменится.
Частое изменение значения *MVLowLmt* или *MVUpLmt* может отрицательно повлиять на качество регулирования и сделать характеристики регулирования неудовлетворительными.
Прежде чем повторно изменять параметр *MVLowLmt* или *MVUpLmt* во время работы, убедитесь, что это не ухудшит качество регулирования.

● ManResetVal (значение ручного сброса)

Эта переменная для данной команды не используется. Любое установленное значение игнорируется.

● MVTrackSw (переключатель отслеживания управляющего воздействия)

Функция отслеживания управляющего воздействия записывает в переменную *MV* внешнее входное значение *MVTrackVal* (отслеживаемое значение MV) во время работы ПИД-регулятора в автоматическом режиме.
Отслеживание управляющего воздействия производится, когда значение переменной *MVTrackSw* = ИСТИНА.
Когда значение переменной *MVTrackSw* снова становится равно ЛОЖЬ, переменная *MV* принимает значение переменной *MVTrackVal* в текущем измерительном цикле, а начиная со следующего измерительного цикла в нее записывается результат вычислений ПИД-регулятора нагрева/охлаждения.
Это позволяет избежать резкого изменения значений *MV_Heat* и *MV_Cool*.



*1. Значение *MV_Heat* определяется на основе *MVTrackVal*.
*2. *MV* принимает значение *MVTrackVal*.

● MVTrackVal (отслеживаемое значение управляющего воздействия)

Это значение, которое записывается в переменную *MV* во время работы функции отслеживания управляющего воздействия (MV).
Значение *MVTrackVal* ограничено значениями параметров *MVLowLmt* и *MVUpLmt*.

● StopMV (управляющее воздействие при остановке)

Это значение, которое переменная *MV* принимает, когда значение переменной *Run* = ЛОЖЬ (т. е. когда выполнение этой команды прекращено).

● ErrorMV (управляющее воздействие при ошибке)

Это значение, которое переменная *MV* принимает, когда возникает ошибка (т. е. когда значение *Error* = ИСТИНА).
Если значение *ErrorMV* находится за пределами допустимого диапазона (-320...320), значение *MV* будет равно 0 при возникновении ошибки.

● Alpha (параметр α 2-ПИД регулятора)

Этот параметр задает значение коэффициента для фильтра уставки.
Более подробную информацию см. в пункте *2-ПИД-регулятор с фильтром уставки* на стр. 2-778 раздела, посвященного команде PIDAT.
В общем случае параметр *Alpha* устанавливается равным 0,65.

● ATCalcGain (коэффициент влияния автонастройки)

Эта переменная позволяет задать коэффициент, с которым постоянные ПИД-регулятора, рассчитанные во время автонастройки, применяются в ПИД-регуляторе.
При значении 1,00 применяются непосредственно значения, полученные в ходе автонастройки.
Для достижения большей стабильности значение *ATCalcGain* следует увеличить, а для повышения скорости реакции — уменьшить.

● ATHystrs (гистерезис автонастройки)

Эта переменная позволяет задать гистерезис, используемый в предельном цикле для выполнения автонастройки.
Чем меньше значение *ATHystrs*, тем точнее выполняется автонастройка. Однако в том случае, если регулируемая величина нестабильна и автонастройка не выполняется должным образом, это значение следует увеличить.
Более подробную информацию см. в пункте *Автоматическая настройка* на стр. 2-780 раздела, посвященного команде PIDAT.

● SampTime (период измерения)

Эта переменная позволяет задать минимальный период, с которым работает ПИД-регулятор нагрева/охлаждения.
Дополнительные сведения см. в разделе *Время выполнения ПИД-регулирования нагрева/охлаждения* на стр. 2-816.
Обработка в ПИД-регуляторе нагрева/охлаждения не выполняется, если время, прошедшее с момента последнего выполнения обработки, меньше времени *SampTime*.

● RngLowLmt (нижняя граница входного диапазона) и RngUpLmt (верхняя граница входного диапазона)

С помощью этих переменных можно задать верхнее и нижнее предельные значения для переменных *PV* и *SP*.
Если значение параметра, подключенного к переменной *PV* или *SP*, выйдет за эти пределы, произойдет ошибка.
Значение *RngLowLmt* всегда должно быть меньше значения *RngUpLmt*.

● DirOpr (направление действия)

Эта переменная для данной команды не используется. Любое установленное значение игнорируется.

● CtlPrd_Cool (период регулирования)

Эта переменная устанавливает период регулирования при использовании выхода *MV_Cool* для широтно-импульсного регулирования (ШИР), когда эта команда применяется совместно с командой *TimeProportionalOut* на стр. 2-839. В нее следует передать то же значение, что и в переменную *CtlPrd* (период регулирования) команды TimeProportionalOut.
Если выход *MV_Cool* не используется для широтно-импульсного регулирования, следует задать значение по умолчанию (T#20 с).

● ProportionalBand_Heat и ProportionalBand_Cool (зоны пропорциональности)

Это одна из трех постоянных ПИД-регулятора. Более подробную информацию см. в пункте *Пропорциональное управляющее воздействие (P)* на стр. 2-773 раздела, посвященного команде PIDAT.

С увеличением значения *ProportionalBand_Heat* и *ProportionalBand_Cool* возрастает статическая ошибка. При слишком узкой зоне пропорциональности могут возникать автоколебания.

● IntegrationTime_Heat и IntegrationTime_Cool (постоянные времени интегрирования)

Это одна из трех постоянных ПИД-регулятора. Более подробную информацию см. в пункте *Управляющее воздействие по интегралу (I)* на стр. 2-775 раздела, посвященного команде PIDAT.

Чем больше значение переменной *IntegrationTime_Heat* или *IntegrationTime_Cool*, тем меньше управляющее воздействие по интегралу.

● DerivativeTime_Heat и DerivativeTime_Cool (постоянные времени дифференцирования)

Это одна из трех постоянных ПИД-регулятора. Более подробную информацию см. в пункте *Управляющее воздействие по производной (D)* на стр. 2-776 раздела, посвященного команде PIDAT.

Чем больше значение переменной *DerivativeTime_Heat* или *DerivativeTime_Cool*, тем большее управляющее воздействие по производной.

● ManMV (ручное управляющее воздействие)

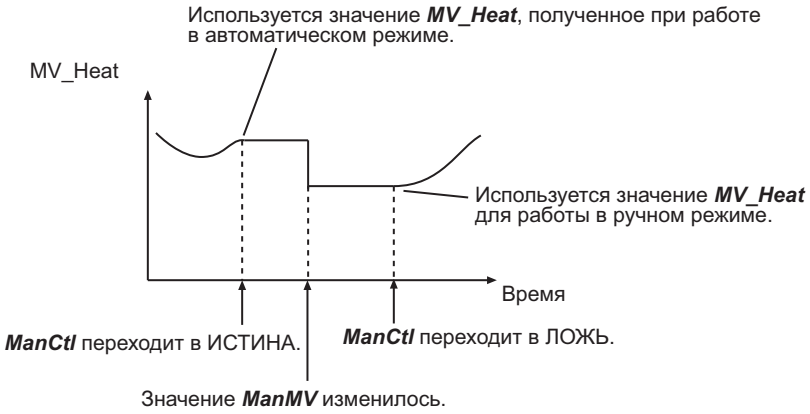
Переменная *MV* принимает это значение во время работы в режиме ручного управления (когда *ManCtl* = ИСТИНА).

Переменная *MV* принимает значение *ManMV*, только если значение *ManMV* изменяется после переключения ПИД-регулятора из автоматического режима в ручной режим.

При переходе из автоматического режима в режим ручного управления значение *MV* становится равно значению *MV_Heat*, если значение *MV_Heat* для работы в автоматическом режиме положительно, или значению *MV_Cool*, если это не так.

Кроме того, после переключения из ручного в автоматический режим значение *MV* становится равно значению *MV_Heat*, если значение *MV_Heat* положительно, или значению *MV_Cool*, если это не так.

Значение *ManMV* не обязательно должно находиться в диапазоне между *MVLowLmt* и *MVUpLmt*.



● ATDone (нормальное завершение автонастройки)

Этот флаг указывает, что автоматическая настройка была завершена нормально.

Он переходит в состояние ИСТИНА, когда автонастройка завершается нормально, и остается в состоянии ИСТИНА до тех пор, пока значение переменной *StartAT* = ИСТИНА.

В указанных ниже случаях этот флаг находится в состоянии ЛОЖЬ.
- Операция автонастройки завершилась с ошибкой.
- Автонастройка выполняется в данный момент (т. е. пока значение *ATBusy* = ИСТИНА).
- В данный момент выполняется ПИД-регулирование нагрева/охлаждения без автонастройки.
- В данный момент ПИД-регулирование нагрева/охлаждения не выполняется (т. е. значение *Run* = ЛОЖЬ).
- Значение переменной *StartAT* = ЛОЖЬ.

● ATBusy (выполнение автонастройки)

Этот флаг указывает на то, что в данный момент выполняется автоматическая настройка. Он находится в состоянии ИСТИНА, пока производится автонастройка. Все остальное время он находится в состоянии ЛОЖЬ.

● MV (управляющее воздействие)

Это величина управляющего воздействия, рассчитанная ПИД-регулятором нагрева/охлаждения. Значения переменных *MV_Heat* и *MV_Cool* определяются в результате преобразования значения *MV*.

● MV_Heat (управляющее воздействие для регулирования нагрева)

Это управляющее воздействие, которое подается на нагревательное устройство.

● MV_Cool (управляющее воздействие для регулирования охлаждения)

Это управляющее воздействие, которое подается на охлаждающее устройство.

ПИД-регулирование нагрева/охлаждения

Подробные сведения о ПИД-регулировании см. в разделе, посвященном команде *PIDAT* на стр. 2-762.
ПИД-регулятор нагрева/охлаждения вычисляет управляющие воздействия (производит «ПИД-обработку»), используя для расчетов значения ПИД-констант для регулирования нагрева и значения ПИД-констант для регулирования охлаждения.
Если значение *MV*, полученное при предыдущей обработке, было меньше или равно 0, используются ПИД-константы для регулирования нагрева. Если предыдущее значение *MV* было больше 0, используются ПИД-константы для регулирования охлаждения.

Пропорциональное (P), интегральное (I) и дифференциальное (D) управление

Подробную информацию о пропорциональном (П), интегральном (И) и дифференциальном (Д) видах регулирования см. в разделе *Пропорциональное (P), интегральное (I) и дифференциальное (D) управление* на стр. 2-773 для команды PIDAT.

2-ПИД-регулятор с фильтром уставки

Подробную информацию о технологии «2-PID» с входным фильтром заданного значения см. в разделе *2-ПИД-регулятор с фильтром уставки* на стр. 2-778 для команды PIDAT.

Запуск ПИД-регулирования

Для выполнения этой команды необходимо использовать подходящие значения постоянных ПИД-регулятора. Возможны два варианта запуска ПИД-регулирования. Выбор соответствующего варианта зависит от того, известны ли к моменту запуска оптимальные значения постоянных ПИД-регулятора.

Значения постоянных ПИД-регулятора можно изменять во время его работы. Также во время работы можно выполнить автоматическую настройку. Для запуска автонастройки во время работы необходимо поменять значение переменной *StartAT* на ИСТИНА.

● Когда оптимальные ПИД-константы неизвестны

В начале работы необходимо выполнить автонастройку для определения оптимальных значений постоянных ПИД-регулятора, если они неизвестны.
Для этого при переходе входа *Run* в состояние ИСТИНА вход *StartAT* должен находиться в состоянии ИСТИНА.
В этом случае сначала выполняется автонастройка, а затем запускается ПИД-регулирование нагрева/охлаждения с использованием найденных значений ПИД-констант.

● Когда оптимальные ПИД-константы известны

Присвойте оптимальные значения постоянных ПИД-регулятора переменным *ProportionalBand_Heat*, *IntegrationTime_Heat*, *DerivativeTime_Heat*, *ProportionalBand_Cool*, *IntegrationTime_Cool* и *DerivativeTime_Cool*, а затем переведите вход *Run* в состояние ИСТИНА. *ProportionalBand_Heat*, *IntegrationTime_Heat*, *DerivativeTime_Heat*, *ProportionalBand_Cool*, *IntegrationTime_Cool* и *DerivativeTime_Cool* являются входными-выходными переменными, поэтому в них нельзя передавать константы в качестве входных параметров. Определите соответствующие переменные и присвойте значения входным параметрам.

Управляющее воздействие в разных состояниях регулятора

Управляющее воздействие (*MV*) определяется в соответствии с текущим состоянием регулятора, что отражено в таблице ниже.

Состояние регулятора	Значение переменной					*MV* (управляющее воздействие)
	ManCtl **(ручное/ автоматическое управление)**	***Run*** **(условие выполнения)**	***Error*** **(завершение с ошибкой)**	***MVTrack Sw*** **(переключатель отслеживания MV)**	***ATBusy*** **(выполнение автонастройки)**	
Завершение с ошибкой	ЛОЖЬ	ИСТИНА	ИСТИНА	---	ЛОЖЬ	*ErrorMV* (управляющее воздействие при ошибке)
Во время работы в автоматическом режиме (отслеживание MV)			ЛОЖЬ	ИСТИНА		*MVTrackVal* (отслеживаемое значение MV)
Во время работы в автоматическом режиме (автонастройка)				ЛОЖЬ	ИСТИНА	Значение многократно изменяется в диапазоне между нижним и верхним предельными значениями MV.
Во время работы в автоматическом режиме (без автонастройки)					ЛОЖЬ	Значение, рассчитанное по текущим значениям ПИД-констант.
Выполнение команды остановлено		ЛОЖЬ	---	---		*StopMV**1 (управляющее воздействие при остановке)
Работа в режиме ручного управления	ИСТИНА	---				*ManMV**2 (ручное управляющее воздействие)

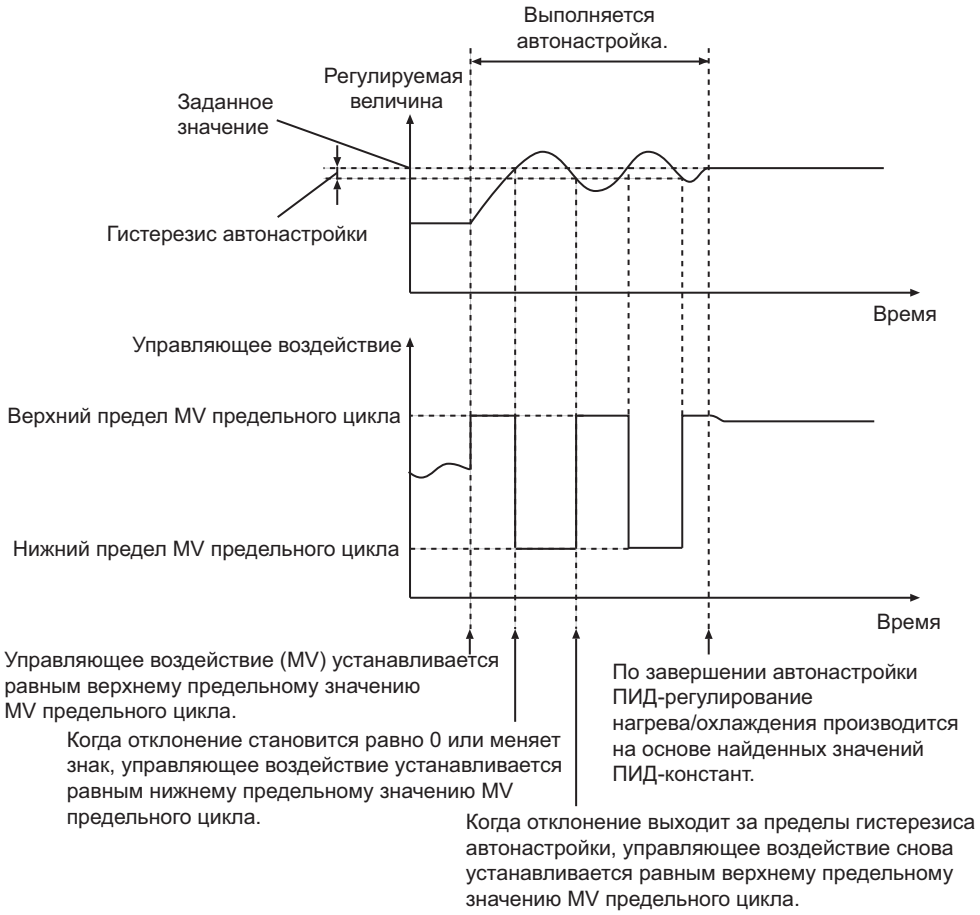
*1. Если значение *StopMV* будет находиться вне допустимого диапазона, *MV* примет значение 0.
*2. Если значение *ManMV* будет находиться вне допустимого диапазона, *MV* примет значение 0.

Автоматическая настройка

Параметр α 2-ПИД регулятора не требуется настраивать часто. Таким образом, основными параметрами, которые настраиваются для этой команды, являются постоянные ПИД-регулятора. Команда PIDAT поддерживает автоматическую настройку постоянных ПИД-регулятора.
Для автонастройки используется метод предельного цикла.
Суть метода предельного цикла состоит в том, что на выходе регулятора временно устанавливаются верхнее и нижнее предельные значения управляющего воздействия предельного цикла, регистрируются соответствующие изменения регулируемой величины и на основе этого определяются оптимальные значения постоянных ПИД-регулятора.

После запуска выполнения автонастройки управляющее воздействие сначала устанавливается равным верхнему предельному значению управляющего воздействия предельного цикла. Когда значение отклонения становится равно 0 или меняет знак, управляющее воздействие устанавливается равным нижнему предельному значению управляющего воздействия предельного цикла. Когда отклонение выходит за пределы гистерезиса автонастройки, управляющее воздействие снова устанавливается равным верхнему предельному значению управляющего

воздействия предельного цикла. Для расчета оптимальных ПИД-констант этот процесс выполняется два с половиной раза.
Верхнее и нижнее предельные значения управляющего воздействия предельного цикла рассчитываются на основе значений параметров.



Автонастройка выполняется во время ПИД-регулирования нагрева/охлаждения, если значение *StartAT* меняется на ИСТИНА во время ПИД-регулирования нагрева/охлаждения (т. е. когда значение *Run* = ИСТИНА). Если же значение *StartAT* уже равно ИСТИНА, когда вход *Run* переходит в состояние ИСТИНА, автонастройка выполняется при запуске ПИД-регулирования.
Рассчитанные значения ПИД-констант применяются сразу после нормального завершения автонастройки.

Если значение переменной *StartAT* поменяется на ЛОЖЬ во время автонастройки (т. е. когда *ATBusy* = ИСТИНА), автонастройка будет отменена. После отмены автонастройки ПИД-регулирование нагрева/охлаждения запускается снова с предыдущими ПИД-константами.

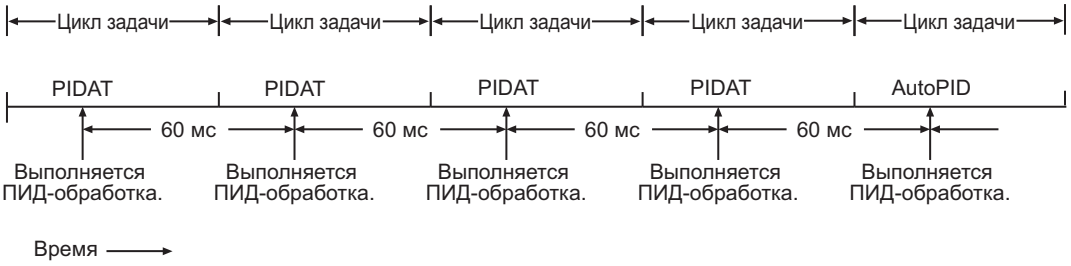
Время выполнения ПИД-регулирования нагрева/охлаждения

ПИД-регулирование нагрева/охлаждения выполняется циклически, с заданным периодом выполнения. Обработка в ПИД-регуляторе нагрева/охлаждения производится, когда в программе пользователя выполняется команда PIDAT.
Однако обработка в ПИД-регуляторе нагрева/охлаждения не производится, если после последнего выполнения обработки прошло меньше времени, чем время *SampTime*.

Если же время, прошедшее с момента последнего выполнения, превышает значение *SampTime*, то излишек времени (прошедшее время - *SampTime*) переносится в следующий период. Более подробная информации дана ниже.
Даже если эта команда не выполняется в результате выполнения команды PrgStop или MC, прошедшее время с момента последнего выполнения обработки в ПИД-регуляторе нагрева/охлаждения обнуляется в моменты времени, обозначенные на рисунках ниже как *Выполняется ПИД-обработка*.

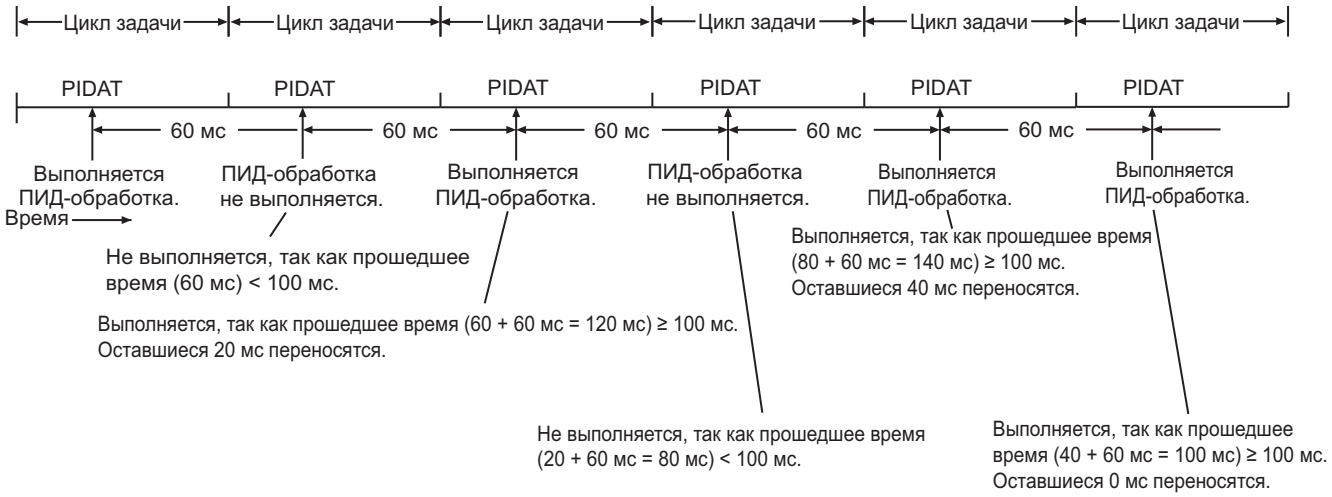
Период выполнения задачи = 60 мс, а ***SampTime*** < 60 мс

Период выполнения задачи больше или равен значению ***SampTime***, поэтому ПИД-обработка выполняется один раз в каждом цикле выполнения задачи.



Период выполнения задачи = 60 мс, а ***SampTime*** =100 мс

Период выполнения задачи меньше значения ***SampTime***, поэтому ПИД-обработка выполняется не в каждом цикле выполнения задачи.



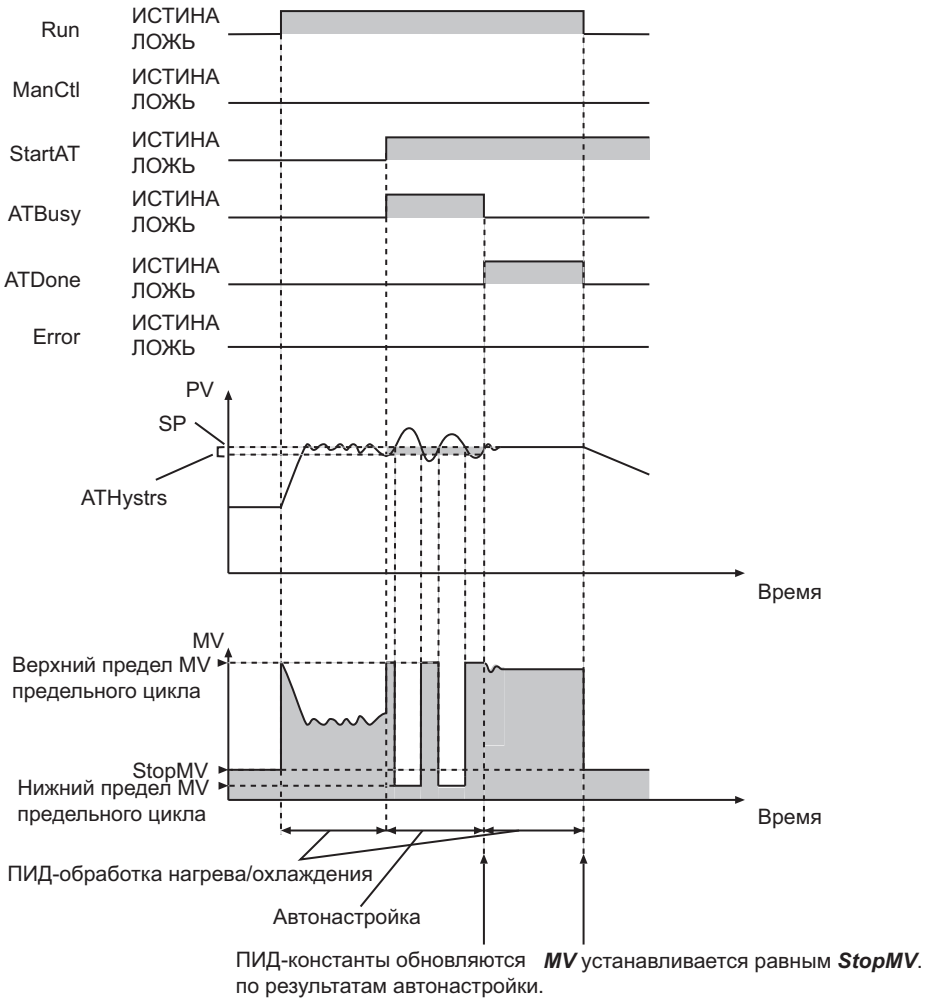
Временные диаграммы

Ниже представлены временные диаграммы изменения переменных команды для разных ситуаций.

● Выполнение автонастройки во время работы в автоматическом режиме

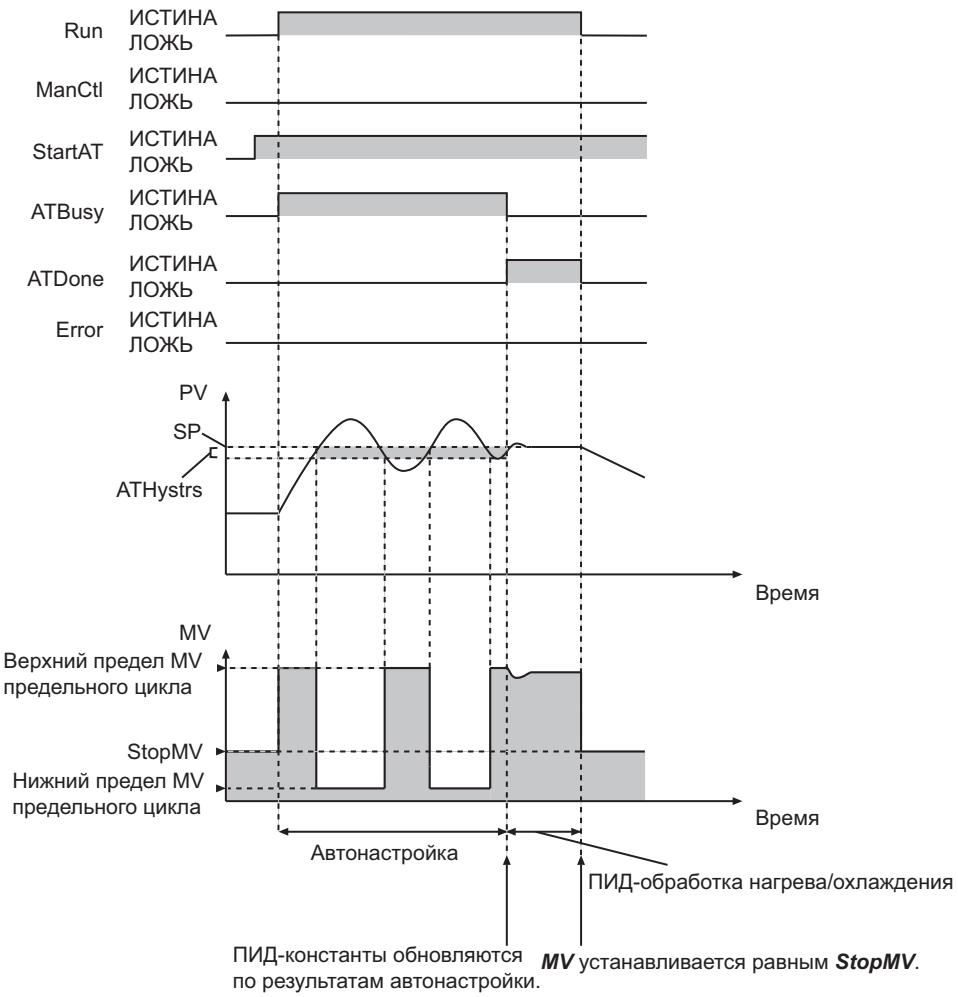
- На приведенном ниже рисунке значение *ManCtl* = ЛОЖЬ, поэтому значение *MV* будет равно *StopMV* все время, пока *Run* = ЛОЖЬ.
- После перехода входа *Run* в состояние ИСТИНА на выход *MV* начинает подаваться значение, рассчитанное на основе постоянных ПИД-регулятора.
- Когда значение *StartAT* меняется на ИСТИНА, запускается автонастройка. Состояние выхода *ATBusy* при этом меняется на ИСТИНА.

- Когда автонастройка завершается, состояние выхода *ATBusy* меняется на ЛОЖЬ, а выход *ATDone* переходит в состояние ИСТИНА.
- По завершении автонастройки управляющее воздействие *MV* рассчитывается на основе постоянных ПИД-регулятора, определенных с помощью автонастройки.
- Когда вход *Run* переходит в состояние ЛОЖЬ, значение *MV* меняется на *StopMV*. После этого выход *ATDone* переходит в состояние ЛОЖЬ.



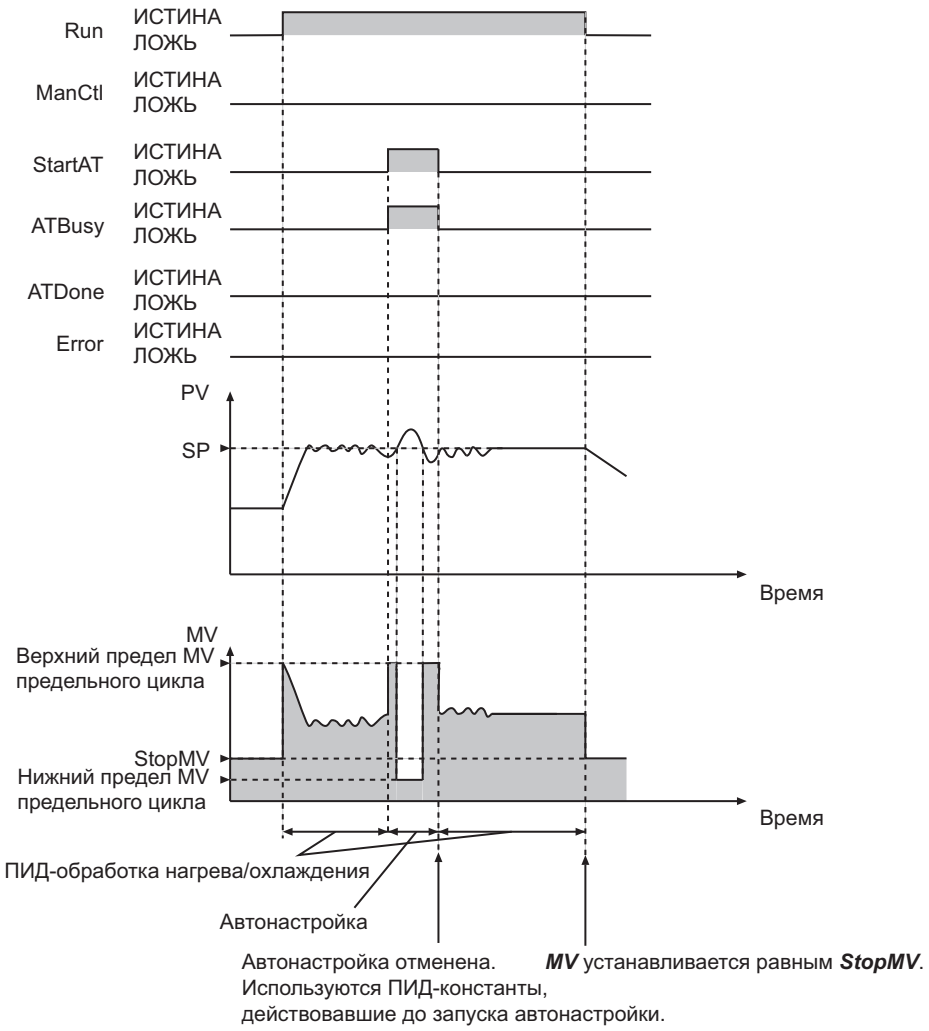
● Выполнение автонастройки в начале выполнения PIDAT

- На приведенном ниже рисунке значение *ManCtl* = ЛОЖЬ, поэтому значение *MV* будет равно *StopMV* все время, пока *Run* = ЛОЖЬ.
- Пока значение *Run* = ЛОЖЬ, автонастройка не будет выполняться, даже если вход *StartAT* перейдет в состояние ИСТИНА.
- Автонастройка запустится, когда оба входа, *StartAT* и *Run*, перейдут в состояние ИСТИНА. Состояние выхода *ATBusy* при этом поменяется на ИСТИНА.
- Когда автонастройка завершается, состояние выхода *ATBusy* меняется на ЛОЖЬ, а выход *ATDone* переходит в состояние ИСТИНА.
- По завершении автонастройки управляющее воздействие *MV* рассчитывается на основе постоянных ПИД-регулятора, определенных с помощью автонастройки.



● Отмена автонастройки

- На приведенном ниже рисунке значение *ManCtl* = ЛОЖЬ, поэтому значение *MV* будет равно *StopMV* все время, пока *Run* = ЛОЖЬ.
- После перехода входа *Run* в состояние ИСТИНА на выход *MV* начинает подаваться значение, рассчитанное на основе постоянных ПИД-регулятора.
- Когда значение *StartAT* меняется на ИСТИНА, запускается автонастройка. Состояние выхода *ATBusy* при этом меняется на ИСТИНА.
- Если значение *StartAT* меняется на ЛОЖЬ во время автонастройки, автонастройка отменяется. Состояние выхода *ATBusy* при этом меняется на ЛОЖЬ.
- После отмены автонастройки выдается управляющее воздействие *MV*, рассчитываемое на основе постоянных ПИД-регулятора, которые использовались непосредственно перед запуском автонастройки.
- Когда вход *Run* переходит в состояние ЛОЖЬ, значение *MV* меняется на *StopMV*.
- Выход *ATDone* не переходит в состояние ИСТИНА, потому что автонастройка была прервана.



● Возникновение ошибки автонастройки во время автонастройки

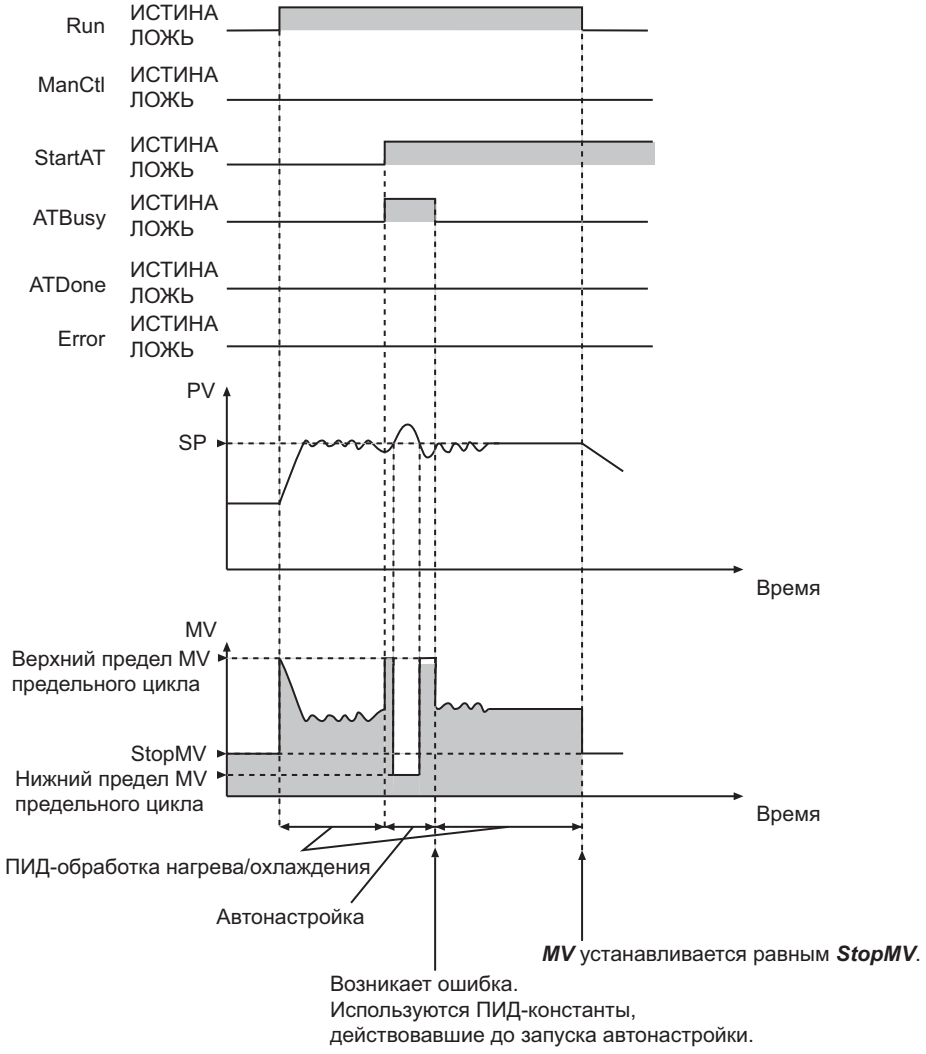
В указанных ниже случаях возникает ошибка автонастройки и автонастройка прекращается.

- Если управляющее воздействие равно верхнему предельному значению управляющего воздействия предельного цикла и время достижения нулевого отклонения превышает 19 999 с.
- Если управляющее воздействие равно нижнему предельному значению управляющего воздействия предельного цикла и время выхода отклонения за пределы гистерезиса (*ATHystrs*) превышает 19 999 с.

Даже если во время автонастройки возникает ошибка, выход *Error* не переходит в состояние ИСТИНА. Автонастройка также не регистрируется в журнале событий.
После отмены автонастройки ПИД-регулирование нагрева/охлаждения запускается снова с предыдущими ПИД-константами.

- На приведенном ниже рисунке значение *ManCtl* = ЛОЖЬ, поэтому значение *MV* будет равно *StopMV* все время, пока *Run* = ЛОЖЬ.
- После перехода входа *Run* в состояние ИСТИНА на выход *MV* начинает подаваться значение, рассчитанное на основе постоянных ПИД-регулятора.
- Когда значение *StartAT* меняется на ИСТИНА, запускается автонастройка. Состояние выхода *ATBusy* при этом меняется на ИСТИНА.
- Если во время выполнения автонастройки возникает ошибка автонастройки, автонастройка немедленно отменяется. Состояние выхода *ATBusy* при этом меняется на ЛОЖЬ.

- Даже если во время автонастройки возникает ошибка, выход *Error* не переходит в состояние ИСТИНА.
- После отмены автонастройки выдается управляющее воздействие *MV*, рассчитываемое на основе постоянных ПИД-регулятора, которые использовались непосредственно перед запуском автонастройки.
- Когда вход *Run* переходит в состояние ЛОЖЬ, значение *MV* меняется на *StopMV*.
- Выход *ATDone* не переходит в состояние ИСТИНА, потому что автонастройка была прервана.



Дополнительная информация

Настройка постоянных ПИД-регулятора

Подробную информацию о способах настройки постоянных ПИД-регулятора см. в разделе *Настройка постоянных ПИД-регулятора* на стр. 2-786 для команды PIDAT.

Начальные значения постоянных ПИД-регулятора для регулирования температуры

Ниже приведены начальные значения постоянных ПИД-регулятора, которые рекомендуется использовать в случае применения команды PIDAT для регулирования температуры. Для других переменных следует использовать значения по умолчанию.

Переменные	Начальные значения (справочно)*1
ProportionalBand_Heat и ProportionalBand_Cool	10 % от полн. диап.
IntegrationTime_Heat и IntegrationTime_Cool	233 с
DerivativeTime_Heat и DerivativeTime_Cool	40 с

*1. При выполнении автонастройки следует использовать результаты, полученные при автонастройке.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Значения переменных *PV* и *SP* должны находиться в диапазоне между значениями *RngLowLmt* и *RngUpLmt* включительно. Единицы измерения этих переменных должны быть согласованы в соответствии с таблицей ниже.

Единица	Значения *PV* и *SP*	Значения *RngLowLmt* и *RngUpLmt*
% от полн. диап.	*PV* = (регулируемая величина в физических единицах - MIN)/(MAX - MIN) × 100*1 *SP* = (заданное значение в физических единицах - MIN)/(MAX - MIN) × 100*1	*RngLowLmt* = 0 *RngUpLmt* = 100
Физическая единица	*PV* = регулируемая величина в физических единицах *SV* = заданное значение в физических единицах	*RngLowLmtl* = MIN*1 *RngUpLmt* = MAX*1

*1. МАКС: верхняя граница входного диапазона в физических единицах измерения; МИН: нижняя граница входного диапазона в физических единицах измерения.

- В следующей таблице показано, изменение каких переменных возможно в том или ином рабочем состоянии.

Переменные	Состояние регулятора		
	Выполнение команды остановлено*1	Работа в автоматическом режиме (автонастройка в данный момент не выполняется)*2	Работа в автоматическом режиме (в данный момент выполняется автонастройка)*3
Run	Возможно	Возможно	Возможно
ManCtl	Возможно	Возможно	Возможно
StartAT	Возможно	Возможно	Возможно
DeadBand	Возможно	Возможно	Возможно
PV	Возможно	Возможно	Возможно
SP	Возможно	Возможно	Невозможно*4
MVLowLmt	Возможно	Возможно	Невозможно*4
MVUpLmt	Возможно	Возможно	Невозможно*4
ManResetVal*5	---	---	---
MVTrackSw	Возможно	Возможно	Невозможно*4
MVTrackVal	Возможно	Возможно	Невозможно*4
StopMV	Возможно	Возможно	Возможно
ErrorMV	Возможно	Возможно	Возможно

Переменные	Состояние регулятора		
	Выполнение команды остановлено*1	Работа в автоматическом режиме (автонастройка в данный момент не выполняется)*2	Работа в автоматическом режиме (в данный момент выполняется автонастройка)*3
Alpha	Возможно	Возможно	Невозможно*4
ATCalcGain	Возможно	Возможно	Невозможно*4
ATHystrs	Возможно	Возможно	Невозможно*4
CtlPrdCool	Возможно	Возможно	Невозможно*4
SampTime	Возможно	Невозможно*6	Невозможно*4
RngLowLmt	Возможно	Невозможно*6	Невозможно*4
RngUpLmt	Возможно	Невозможно*6	Невозможно*4
DirOpr*5	---	---	---
ProportinalBand_Heat	Возможно	Возможно	Невозможно*7
IntegrationTime_Heat	Возможно	Возможно	Невозможно*7
Derivative Time_Heat	Возможно	Возможно	Невозможно*7
ProportinalBand_Cool	Возможно	Возможно	Невозможно*7
IntegrationTime_Cool	Возможно	Возможно	Невозможно*7
Derivative Time_Cool	Возможно	Возможно	Невозможно*7
ManMV	Возможно	Возможно	Возможно

*1. *ManCtl* = ИСТИНА, *Run* = ЛОЖЬ, *Error* = ИСТИНА или *MVTrackSw* = ИСТИНА.
*2. *ManCtl* = ЛОЖЬ, *Run* = ИСТИНА, *Error* = ЛОЖЬ, *MVTrackSw* = ЛОЖЬ и *ATBusy* = ЛОЖЬ.
*3. *ManCtl* = ЛОЖЬ, *Run* = ИСТИНА, *Error* = ЛОЖЬ, *MVTrackSw* = ЛОЖЬ и *ATBusy* = ИСТИНА.
*4. Автонастройка выполняется с использованием значения, действовавшего непосредственно перед выполнением автонастройки.
*5. Эта переменная для данной команды не используется. Это значение можно изменить, но оно игнорируется.
*6. Операция выполняется с использованием значения, действовавшего непосредственно перед выполнением операции.
*7. Это значение можно изменить, но оно игнорируется. По завершении автонастройки текущие значения заменяются значениями, рассчитанными при автонастройке.

- Значение *SampTime* усекается с точностью до 100 нс.
- Если значение *StartAT* поменяется на ИСТИНА, когда значение *ManCtl* = ИСТИНА, автонастройка запустится в следующий раз, когда значение *ManCtl* поменяется на ЛОЖЬ.
- Если значение *ErrorMV* находится за пределами допустимого диапазона (-320...320), значение *MV* будет равно 0 при возникновении ошибки.
- Если значение *ManCtl* поменяется на ИСТИНА во время автонастройки, автонастройка будет отменена.
- Даже если во время автонастройки возникает ошибка, выход *Error* не переходит в состояние ИСТИНА. Автонастройка не регистрируется в журнале событий.
- В указанном ниже случае происходит ошибка.
  Выход *Error* перейдет в состояние ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки. Выходы *ATDone* и *ATBusy* перейдут в состояние ЛОЖЬ.
  Если входы *ManCtl* и *Run* будут находиться в состоянии ЛОЖЬ, переменная *MV* примет значение *ErrorMV*. Если значение *ErrorMV* будет находиться вне допустимого диапазона, *MV* примет значение 0.

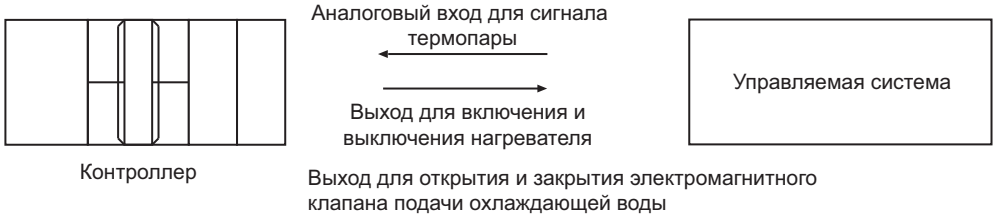
Ошибка	Значение *ErrorID*
Значение входной переменной находится за пределами допустимого диапазона.	16#0400

Ошибка	Значение *ErrorID*
RngLowLmt больше или равно *RngUpLmt*.	16#0401
MVLowLmt больше или равно *MVUpLmt*.	

- Если работа регулятора должна прекращаться при ошибках, не указанных выше, в программе системы необходимо предусмотреть, чтобы значение переменной *Run* менялось на ЛОЖЬ при возникновении ошибки.
- Если возникает ошибка из-за того, что значение *PV* или *SP* вышло за допустимый диапазон, состояние ошибки длится не менее 5 секунд, даже если значение возвращается в границы допустимого диапазона раньше. Это означает, что выход *Error* будет находиться в состоянии ЛОЖЬ пять секунд.
- Если после сброса ошибки вход *Run* находится в состоянии ИСТИНА, ПИД-регулирование нагрева/охлаждения автоматически перезапускается. Если кроме входа *Run* в состоянии ИСТИНА также находится вход *StartAT*, также автоматически перезапускается автонастройка.
- Проверка на наличие ошибок производится в каждом цикле измерения.
- Если соблюдаются указанные ниже условия и выполняются операции резервного копирования и восстановления, значения ПИД-констант, полученные путем автонастройки, вернутся к предыдущим значениям, которые были рассчитаны до операции резервного копирования. Принимайте эту особенность во внимание.
  a) Для входных-выходных параметров указан атрибут Retain («Сохранение»).
  b) Операции выполняются в следующем порядке: резервное копирование, автонастройка, затем восстановление.
- При переходе из режима автоматической работы в ручной режим используется то из значений *MV_Heat* и *MV_Cool*, которое является положительным, что позволяет избежать резких изменений в процессе («безударное» переключение). Однако вследствие этого значение другой переменной может резко измениться.

Пример программы

Рассмотрим пример программы, в которой для регулирования температуры применяется команда PIDAT_HeatCool.
Используется один аналоговый вход для приема сигнала термопары, установленной в управляемой системе.
Также используются два дискретных выхода для подачи сигналов в управляемую систему: один сигнал для управления нагревом и один сигнал для управления охлаждением.
Дискретный выход нагрева включает или выключает нагревательное устройство. Дискретный выход охлаждения закрывает или открывает электромагнитный клапан подачи охлаждающей воды.



Конфигурация модулей

Подключены указанные ниже модули.

- CJ1W-AD04U (модуль универсальных входов с гальванической развязкой)

- CJ1W-OC201 (модуль релейных выходов)

Карта входов-выходов

Карты входов-выходов для модулей настроены так, как показано в следующих таблицах.

● CJ1W-AD04U

Порт	Описание	Чтение/запись	Тип данных	Переменная	Комментарий к переменной	Тип переменной
Ch1_AIInPV	Вход 1, регулируемая величина	Чт.	INT	J01_Ch1_AIInPV	Вход термопары	Глобальная переменная

● CJ1W-OC201

Порт	Описание	Чтение/запись	Тип данных	Переменная	Комментарий к переменной	Тип переменной
Ch1_Out00	Бит 00 выходного слова 1	Чт./зап.	BOOL	J02_Ch1_Out00	Выход на нагреватель	Глобальная переменная
Ch1_Out04	Бит 04 выходного слова 1	Чт./зап.	BOOL	J02_Ch1_Out04	Выход на охлаждающее устройство	Глобальная переменная

Описание сенсорной панели

В этом примере предполагается, что сенсорная панель подключена к контроллеру. Ниже указаны входные и выходные данные, доступные для просмотра и управления на экране сенсорной панели.

Вход/выход	Информация
Входы	Флаг выполнения программы-примера Флаг ручного/автоматического управления Заданное значение Флаг выполнения автонастройки Зона нечувствительности Начальные параметры Рабочие параметры
Вход/выход	Зона пропорциональности, постоянная времени интегрирования и постоянная времени дифференцирования для регулирования нагрева Зона пропорциональности, постоянная времени интегрирования и постоянная времени дифференцирования для регулирования охлаждения Ручное управляющее воздействие

Вход/выход	Информация
Выходы	Регулируемая величина Флаг нормального завершения автонастройки Флаг выполнения автонастройки Флаг ошибки Управляющее воздействие Управляющее воздействие для регулирования нагрева Управляющее воздействие для регулирования охлаждения

Преобразование управляющих воздействий в сигналы ШИР

В данном примере и для нагревательного, и для охлаждающего устройств используются дискретные выходы, которые могут находиться в одном из двух состояний: ВКЛ или ВЫКЛ. Следовательно, управляющие воздействия для нагревательного и охлаждающего устройств необходимо преобразовать в сигналы для широтно-импульсного регулирования (ШИР).
Такое преобразование выполняет команда *TimeProportionalOut* на стр. 2-839.
Однако во время автонастройки выходные сигналы, подаваемые на нагревательное и охлаждающее устройства, должны изменяться сразу после изменения значений выходов *MV_Heat* и *MV_Cool* команды PIDAT_HeatCool. Поэтому команду TimeProportionalOut использовать нельзя. Если бы использовалась команда TimeProportionalOut, то состояния выходов на нагревательное и охлаждающее устройства изменялись бы только с периодом регулирования, заданным пользователем. Поэтому в данном примере для преобразования управляющих воздействий в выходные сигналы ШИР во время автонастройки используются команды таймера.

Определения глобальных переменных

● Глобальные переменные

Переменная	Тип данных	Начальное значение	AT	Сохранение	Публикация в сети	Комментарий
J01_Ch1_AIInPV	INT	0	IOBus://rack#0/slot#0/Ch1_AIInPV	☐	Не публикуется	Вход сигнала термопары от CJ1W-AD04U
J02_Ch1_Out00	BOOL	ЛОЖЬ	IOBus://rack#0/slot#1/Ch1_Out/Ch1_Out00	☐	Не публикуется	Выход сигнала нагрева на CJ1W-OC201
J02_Ch1_Out04	BOOL	ЛОЖЬ	IOBus://rack#0/slot#1/Ch1_Out/Ch1_Out04	☐	Не публикуется	Выход сигнала охлаждения на CJ1W-OC201
PTIn_Run	BOOL	ЛОЖЬ		☑	Вход	Ввод флага выполнения программы-примера с сенсорной панели
PTIn_ManCtl	BOOL	ЛОЖЬ		☑	Вход	Ввод флага ручного/автоматического управления с сенсорной панели
PTIn_SP	REAL			☑	Вход	Ввод заданного значения с сенсорной панели
PTIn_StartAT	BOOL	ЛОЖЬ		☑	Вход	Ввод флага выполнения автонастройки с сенсорной панели

Переменная	Тип данных	Начальное значение	AT	Сохранение	Публикация в сети	Комментарий
PTIn_DeadBand	REAL	0		☑	Вход	Ввод зоны нечувствительности с сенсорной панели
PTIn_InitParam	_sINIT_SET_PARAMS	(SampTime := T#100 ms, RngLowLmt := 0.0, RngUpLmt := 100.0, DirOpr := False)		☑	Вход	Ввод начальных параметров с сенсорной панели
PTIn_InitSetOpr_SampTime	LINT	100		☑	Вход	Ввод периода измерения с сенсорной панели (единицы: мс)
PTIn_OprParam	_sOPR_SET_PARAMS	(MVLowLmt := -100, MVUpLmt := 100, ManResetVal := 0.0, MVTrackSw := False, MVTrackVal := 0.0, StopMV := 0.0, ErrorMV := 0.0,Alpha := 0.65, ATCalcGain := 1.0, ATHystrs := 0.2)		☑	Вход	Ввод рабочих параметров с сенсорной панели
PTOut_PV	REAL	0		☐	Выход	Вывод регулируемой величины на сенсорную панель
PT_PB_Heat	REAL	1		☑	Вход	Ввод/вывод зоны пропорциональности для регулирования нагрева с/на сенсорную панель
PT_TI_Heat	LINT	1000		☑	Вход	Ввод/вывод постоянной времени интегрирования для регулирования нагрева с/на сенсорную панель (единицы: мс)
PT_TD_Heat	LINT	1000		☑	Вход	Ввод/вывод постоянной времени дифференцирования для регулирования нагрева с/на сенсорную панель (единицы: мс)
PT_PB_Cool	REAL	1		☑	Вход	Ввод/вывод зоны пропорциональности для регулирования охлаждения с/на сенсорную панель
PT_TI_Cool	LINT	1000		☑	Вход	Ввод/вывод постоянной времени интегрирования для регулирования охлаждения с/на сенсорную панель (единицы: мс)
PT_TD_Cool	LINT	1000		☑	Вход	Ввод/вывод постоянной времени дифференцирования для регулирования охлаждения с/на сенсорную панель (единицы: мс)

Переменная	Тип данных	Начальное значение	AT	Сохранение	Публикация в сети	Комментарий
PT_ManMV	REAL	0		☑	Вход	Ввод/вывод ручного управляющего воздействия с/на сенсорную панель
PTOut_ATDone	BOOL	ЛОЖЬ		☐	Выход	Вывод флага нормального завершения автонастройки на сенсорную панель
PTOut_ATBusy	BOOL	ЛОЖЬ		☐	Выход	Вывод флага выполнения автонастройки на сенсорную панель
PTOut_Error	BOOL	ЛОЖЬ		☐	Выход	Вывод флага ошибки на сенсорную панель
PTOut_MV	REAL	0		☐	Выход	Вывод управляющего воздействия на сенсорную панель
PTOut_MVHeat	REAL	0		☐	Выход	Вывод управляющего воздействия для регулирования нагрева на сенсорную панель
PTOut_MVCool	REAL	0		☐	Выход	Вывод управляющего воздействия для регулирования охлаждения на сенсорную панель

Программа на языке LD

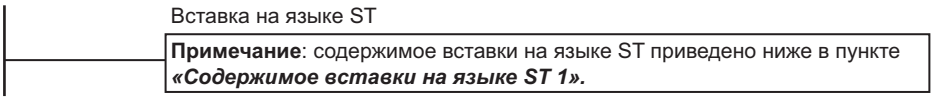
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	PB_Heat	REAL	0	Зона пропорциональности для регулирования нагрева
	PB_Cool	REAL	0	Зона пропорциональности для регулирования охлаждения
	MV	REAL	0	Управляющее воздействие
	MV_Heat	REAL	0	Управляющее воздействие для регулирования нагрева
	MV_Cool	REAL	0	Управляющее воздействие для регулирования охлаждения
	PIDAT_HeatCool_inst	PIDAT_HeatCool		Экземпляр команды PIDAT_HeatCool
	TI_Heat	TIME	T#0 c	Постоянная времени интегрирования для регулирования нагрева
	TI_Cool	TIME	T#0 c	Постоянная времени интегрирования для регулирования охлаждения
	TD_Heat	TIME	T#0 c	Постоянная времени дифференцирования для регулирования нагрева
	TD_Cool	TIME	T#0 c	Постоянная времени дифференцирования для регулирования охлаждения
	ManMV	REAL	0	Ручное управляющее воздействие

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	CtlPrd_Cool	TIME	T#20 c	Период регулирования охлаждения
	CtlPrd_Heat	TIME	T#2 c	Период регулирования нагрева
	TPOHeat_inst	TimeProportionalOut		Экземпляр команды TimeProportionalOut для регулирования нагрева
	TPOCool_inst	TimeProportionalOut		Экземпляр команды TimeProportionalOut для регулирования охлаждения
	ATHeatPhase	BOOL	ЛОЖЬ	Флаг автонастройки для регулирования нагрева
	ATCoolPhase	BOOL	ЛОЖЬ	Флаг автонастройки для регулирования охлаждения
	MVHeatTime	TIME	T#0 c	Время автонастройки для регулирования нагрева
	MVCoolTime	TIME	T#0 c	Время автонастройки для регулирования охлаждения
	AT_Heat_inst	TP		Экземпляр команды TP для вывода управляющего воздействия для регулирования нагрева во время автонастройки
	AT_Cool_inst	TP		Экземпляр команды TP для вывода управляющего воздействия для регулирования охлаждения во время автонастройки
	EachCtlPrd_ATHeat_inst	TON		Экземпляр команды TON для вывода управляющего воздействия для регулирования нагрева во время автонастройки
	EachCtlPrd_ATCool_inst	TON		Экземпляр команды TON для вывода управляющего воздействия для регулирования охлаждения во время автонастройки
	PV	REAL	0	Регулируемая величина

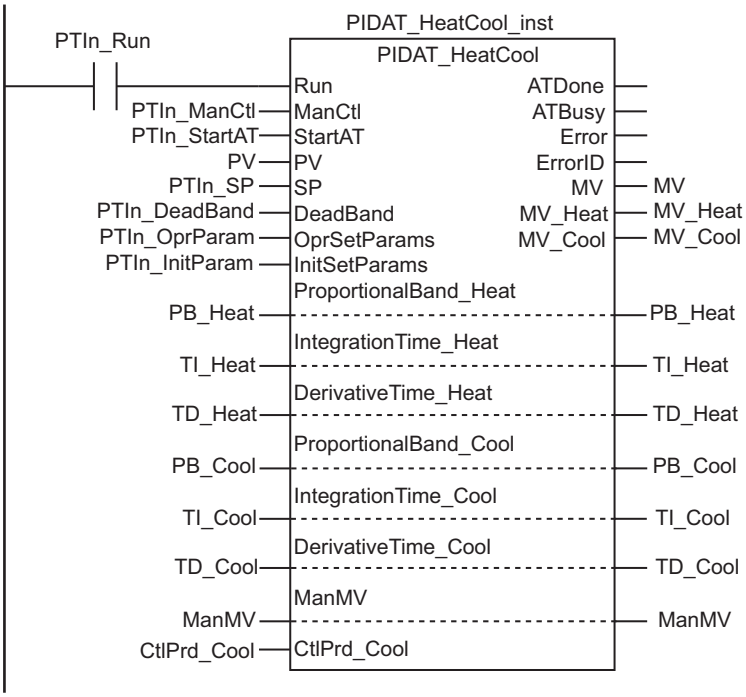
Внешние переменные	Переменная	Тип данных	Комментарий
	J01_Ch1_AIInPV	INT	Вход сигнала термопары от CJ1W-AD04U
	J02_Ch1_Out00	BOOL	Выход сигнала нагрева на CJ1W-OC201
	J02_Ch1_Out04	BOOL	Выход сигнала охлаждения на CJ1W-OC201
	PTIn_Run	BOOL	Ввод флага выполнения программы-примера с сенсорной панели
	PTIn_ManCtl	BOOL	Ввод флага ручного/автоматического управления с сенсорной панели
	PTIn_SP	REAL	Ввод заданного значения с сенсорной панели
	PTIn_StartAT	BOOL	Ввод флага выполнения автонастройки с сенсорной панели

Внешние переменные	Переменная	Тип данных	Комментарий
	PTIn_DeadBand	REAL	Ввод зоны нечувствительности с сенсорной панели
	PTIn_InitParam	_sINIT_SET_PARAMS	Ввод начальных параметров с сенсорной панели
	PTIn_InitSetOpr_SampTime	LINT	Ввод периода измерения с сенсорной панели (единицы: мс)
	PTIn_OprParam	_sOPR_SET_PARAMS	Ввод рабочих параметров с сенсорной панели
	PTOut_PV	REAL	Вывод регулируемой величины на сенсорную панель
	PT_PB_Heat	REAL	Ввод/вывод зоны пропорциональности для регулирования нагрева с/на сенсорную панель
	PT_TI_Heat	LINT	Ввод/вывод постоянной времени интегрирования для регулирования нагрева с/на сенсорную панель (единицы: мс)
	PT_TD_Heat	LINT	Ввод/вывод постоянной времени дифференцирования для регулирования нагрева с/на сенсорную панель (единицы: мс)
	PT_PB_Cool	REAL	Ввод/вывод зоны пропорциональности для регулирования охлаждения с/на сенсорную панель
	PT_TI_Cool	LINT	Ввод/вывод постоянной времени интегрирования для регулирования охлаждения с/на сенсорную панель (единицы: мс)
	PT_TD_Cool	LINT	Ввод/вывод постоянной времени дифференцирования для регулирования охлаждения с/на сенсорную панель (единицы: мс)
	PT_ManMV	REAL	Ввод/вывод ручного управляющего воздействия с/на сенсорную панель
	PTOut_ATDone	BOOL	Вывод флага нормального завершения автонастройки на сенсорную панель
	PTOut_ATBusy	BOOL	Вывод флага выполнения автонастройки на сенсорную панель
	PTOut_Error	BOOL	Вывод флага ошибки на сенсорную панель
	PTOut_MV	REAL	Вывод управляющего воздействия на сенсорную панель
	PTOut_MVHeat	REAL	Вывод управляющего воздействия для регулирования нагрева на сенсорную панель
	PTOut_MVCool	REAL	Вывод управляющего воздействия для регулирования охлаждения на сенсорную панель

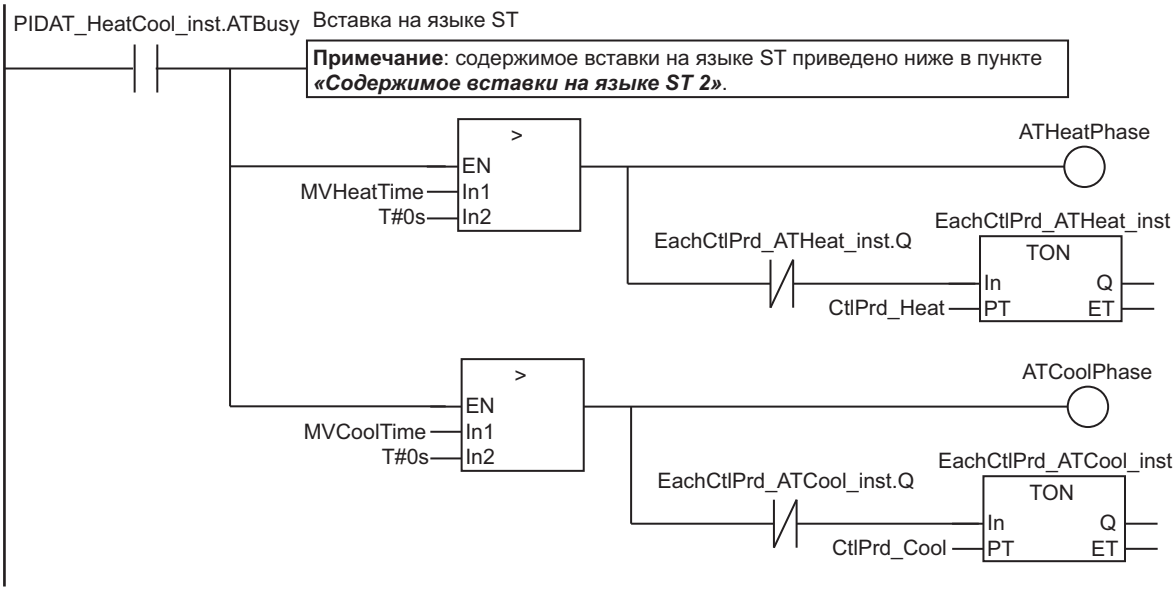
Преобразование единиц ввода значений от CJ1W-AD04U и с сенсорной панели.



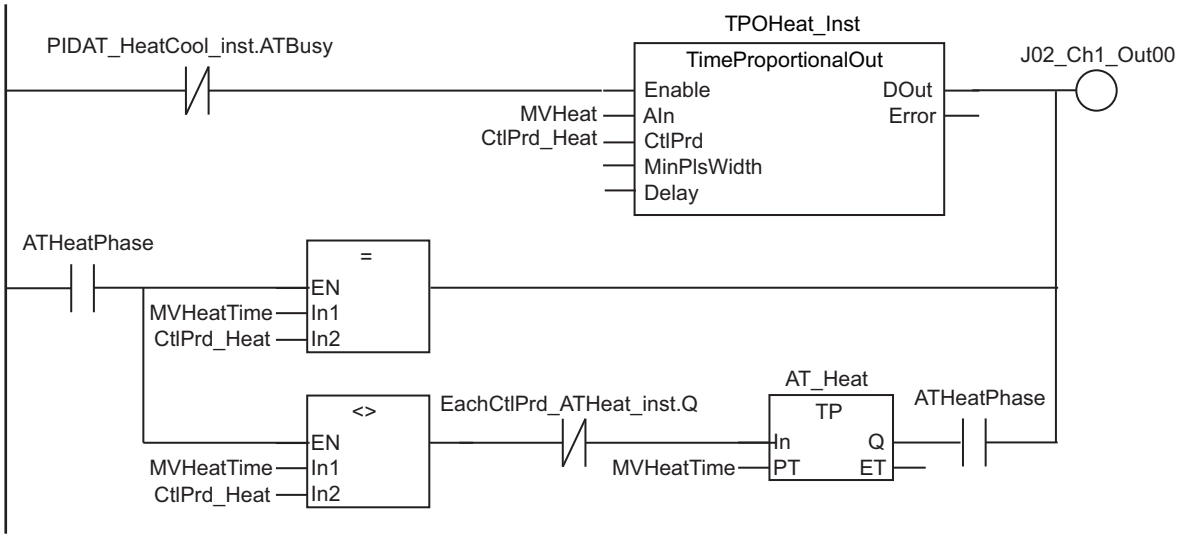
Выполнение команды PIDAT_HeatCool.



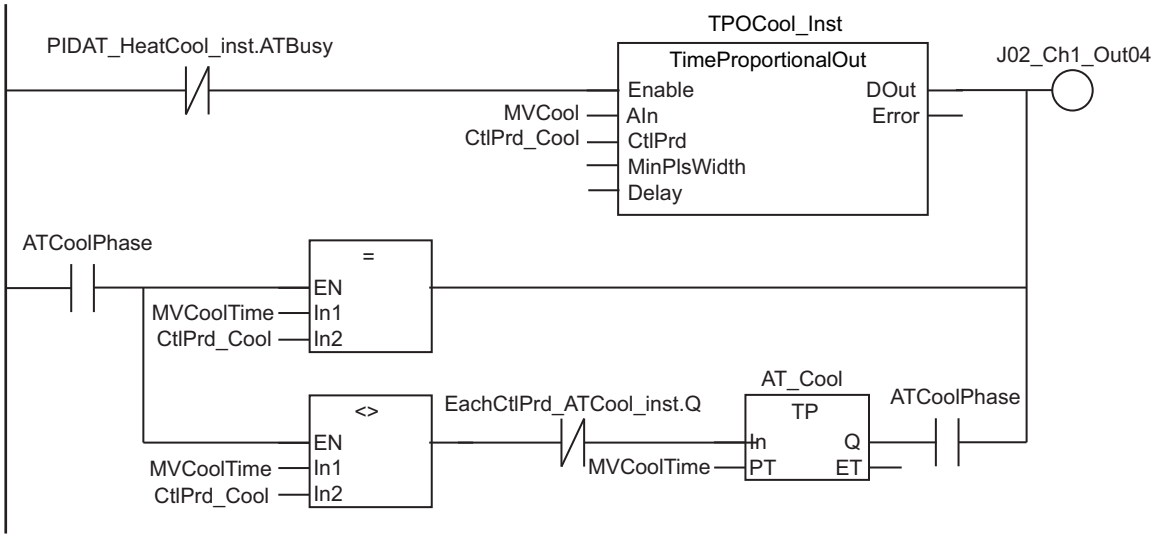
Подготовка к преобразованию значений MV в сигналы ШИР во время выполнения автонастройки.



Выдача сигнала нагрева на модуль CJ1W-OC201



Выдача сигнала охлаждения на модуль CJ1W-OC201



Получение значений для вывода на сенсорную панель.

Вставка на языке ST

Примечание: содержимое вставки на языке ST приведено ниже в пункте ***«Содержимое вставки на языке ST 3».***

● Содержимое вставки на языке ST 1

```
// Преобразование единиц ввода значений от CJ1W-AD04U и с сенсорной панели.
PV := INT_TO_REAL(J01_Ch1_AIInPV)/REAL#10.0;

PTIn_InitParam.SampTime := NanoSecToTime(PTIn_InitSetOpr_SampTime*1000000);

PB_Heat := PT_PB_Heat;
TI_Heat := NanoSecToTime(PT_TI_Heat*1000000);
TD_Heat := NanoSecToTime(PT_TD_Heat*1000000);
PB_Cool := PT_PB_Cool;
TI_Cool := NanoSecToTime(PT_TI_Cool*1000000);
TD_Cool := NanoSecToTime(PT_TD_Cool*1000000);
```

```
ManMV := PT_ManMV;
```

● Содержимое вставки на языке ST 2

```
MVHeatTime := MULTIME(CtlPrd_Heat,(MV_Heat/100));
MVCoolTime := MULTIME(CtlPrd_Cool,(MV_Cool/100));
```

● Содержимое вставки на языке ST 3

```
// Получение значений для вывода на сенсорную панель.
PTOut_PV := PV;

PTOut_ATDone := PIDAT_HeatCool_inst.ATDone;
PTOut_ATBusy := PIDAT_HeatCool_inst.ATBusy;
PTOut_Error := PIDAT_HeatCool_inst.Error;

PTOut_MV := PIDAT_HeatCool_inst.MV;
PTOut_MVHeat := PIDAT_HeatCool_inst.MV_Heat;
PTOut_MVCool := PIDAT_HeatCool_inst.MV_Cool;

PT_PB_Heat := PB_Heat;
PT_TI_Heat := TimeToNanoSec( TI_Heat )/1000000;
PT_TD_Heat := TimeToNanoSec( TD_Heat )/1000000;

PT_PB_Cool := PB_Cool;
PT_TI_Cool := TimeToNanoSec( TI_Cool )/1000000;
PT_TD_Cool := TimeToNanoSec( TD_Cool )/1000000;

PT_ManMV := ManMV;
```

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	PB_Heat	REAL	0	Зона пропорциональности для регулирования нагрева
	PB_Cool	REAL	0	Зона пропорциональности для регулирования охлаждения
	MV	REAL	0	Управляющее воздействие
	MV_Heat	REAL	0	Управляющее воздействие для регулирования нагрева
	MV_Cool	REAL	0	Управляющее воздействие для регулирования охлаждения
	PIDAT_HeatCool_inst	PIDAT_HeatCool		Экземпляр команды PIDAT_HeatCool
	TI_Heat	TIME	T#0 c	Постоянная времени интегрирования для регулирования нагрева
	TI_Cool	TIME	T#0 c	Постоянная времени интегрирования для регулирования охлаждения

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	TD_Heat	TIME	T#0 c	Постоянная времени дифференцирования для регулирования нагрева
	TD_Cool	TIME	T#0 c	Постоянная времени дифференцирования для регулирования охлаждения
	ManMV	REAL	0	Ручное управляющее воздействие
	CtlPrd_Cool	TIME	T#20 c	Период регулирования охлаждения
	CtlPrd_Heat	TIME	T#2 c	Период регулирования нагрева
	TPOHeat_inst	TimeProportionalOut		Экземпляр команды TimeProportionalOut для регулирования нагрева
	TPOCool_inst	TimeProportionalOut		Экземпляр команды TimeProportionalOut для регулирования охлаждения
	ATHeatPhase	BOOL	ЛОЖЬ	Флаг автонастройки для регулирования нагрева
	ATCoolPhase	BOOL	ЛОЖЬ	Флаг автонастройки для регулирования охлаждения
	MVHeatTime	TIME	T#0 c	Время автонастройки для регулирования нагрева
	MVCoolTime	TIME	T#0 c	Время автонастройки для регулирования охлаждения
	AT_Heat_inst	TP		Экземпляр команды TP для вывода управляющего воздействия для регулирования нагрева во время автонастройки
	AT_Cool_inst	TP		Экземпляр команды TP для вывода управляющего воздействия для регулирования охлаждения во время автонастройки
	EachCtlPrd_ATHeat_inst	TON		Экземпляр команды TON для вывода управляющего воздействия для регулирования нагрева во время автонастройки
	EachCtlPrd_ATCool_inst	TON		Экземпляр команды TON для вывода управляющего воздействия для регулирования охлаждения во время автонастройки
	PV	REAL	0	Регулируемая величина

Внешние переменные	Переменная	Тип данных	Комментарий
	J01_Ch1_AIInPV	INT	Вход сигнала термопары от CJ1W-AD04U
	J02_Ch1_Out00	BOOL	Выход сигнала нагрева на CJ1W-OC201
	J02_Ch1_Out04	BOOL	Выход сигнала охлаждения на CJ1W-OC201
	PTIn_Run	BOOL	Ввод флага выполнения программы-примера с сенсорной панели

Внешние переменные	Переменная	Тип данных	Комментарий
	PTIn_ManCtl	BOOL	Ввод флага ручного/автоматического управления с сенсорной панели
	PTIn_SP	REAL	Ввод заданного значения с сенсорной панели
	PTIn_StartAT	BOOL	Ввод флага выполнения автонастройки с сенсорной панели
	PTIn_DeadBand	REAL	Ввод зоны нечувствительности с сенсорной панели
	PTIn_InitParam	_sINIT_SET_PARAMS	Ввод начальных параметров с сенсорной панели
	PTIn_InitSetOpr_SampTime	LINT	Ввод периода измерения с сенсорной панели (единицы: мс)
	PTIn_OprParam	_sOPR_SET_PARAMS	Ввод рабочих параметров с сенсорной панели
	PTOut_PV	REAL	Вывод регулируемой величины на сенсорную панель
	PT_PB_Heat	REAL	Ввод/вывод зоны пропорциональности для регулирования нагрева с/на сенсорную панель
	PT_TI_Heat	LINT	Ввод/вывод постоянной времени интегрирования для регулирования нагрева с/на сенсорную панель (единицы: мс)
	PT_TD_Heat	LINT	Ввод/вывод постоянной времени дифференцирования для регулирования нагрева с/на сенсорную панель (единицы: мс)
	PT_PB_Cool	REAL	Ввод/вывод зоны пропорциональности для регулирования охлаждения с/на сенсорную панель
	PT_TI_Cool	LINT	Ввод/вывод постоянной времени интегрирования для регулирования охлаждения с/на сенсорную панель (единицы: мс)
	PT_TD_Cool	LINT	Ввод/вывод постоянной времени дифференцирования для регулирования охлаждения с/на сенсорную панель (единицы: мс)
	PT_ManMV	REAL	Ввод/вывод ручного управляющего воздействия с/на сенсорную панель
	PTOut_ATDone	BOOL	Вывод флага нормального завершения автонастройки на сенсорную панель
	PTOut_ATBusy	BOOL	Вывод флага выполнения автонастройки на сенсорную панель
	PTOut_Error	BOOL	Вывод флага ошибки на сенсорную панель
	PTOut_MV	REAL	Вывод управляющего воздействия на сенсорную панель

Внешние переменные	Переменная	Тип данных	Комментарий
	PTOut_MVHeat	REAL	Вывод управляющего воздействия для регулирования нагрева на сенсорную панель
	PTOut_MVCool	REAL	Вывод управляющего воздействия для регулирования охлаждения на сенсорную панель

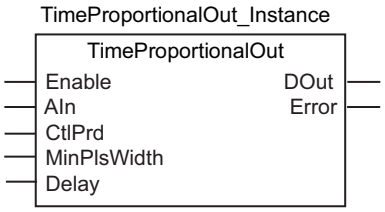
```
// Преобразование единиц ввода значений от CJ1W-AD04U и с сенсорной панели.
PV := INT_TO_REAL(J01_Ch1_AIInPV)/REAL#10.0;

PTIn_InitParam.SampTime := NanoSecToTime(PTIn_InitSetOpr_SampTime*1000000);

PB_Heat := PT_PB_Heat;
TI_Heat := NanoSecToTime(PT_TI_Heat*1000000);
TD_Heat := NanoSecToTime(PT_TD_Heat*1000000);

PB_Cool := PT_PB_Cool;
TI_Cool := NanoSecToTime(PT_TI_Cool*1000000);
TD_Cool := NanoSecToTime(PT_TD_Cool*1000000);

ManMV := PT_ManMV;

// Выполнение команды PIDAT_HeatCool.
PIDAT_HeatCool_inst(Run :=PTIn_Run,
  ManCtl :=PTIn_ManCtl,
  StartAT :=PTIn_StartAT,
  PV :=PV,
  SP :=PTIn_SP,
  DeadBand :=PTIn_DeadBand,
  OprSetParams :=PTIn_OprParam,
  InitSetParams :=PTIn_InitParam,
  ProportionalBand_Heat :=PB_Heat,
  IntegrationTime_Heat :=TI_Heat,
  DerivativeTime_Heat :=TD_Heat,
  ProportionalBand_Cool :=PB_Cool,
  IntegrationTime_Cool :=TI_Cool,
  DerivativeTime_Cool :=TD_Cool,
  ManMV :=ManMV,
  CtlPrd_Cool :=CtlPrd_Cool,
  MV =>MV,
  MV_Heat =>MV_Heat,
  MV_Cool =>MV_Cool);

// Подготовка к преобразованию значений MV в сигналы ШИР во время выполнения автона
стройки.
IF PIDAT_HeatCool_inst.ATBusy THEN
  MVHeatTime := MULTIME(CtlPrd_Heat, (MV_Heat/100) );
```

```
  MVCoolTime := MULTIME(CtlPrd_Cool, (MV_Cool/100) );
END_IF;

ATHeatPhase := PIDAT_HeatCool_inst.ATBusy & (MVHeatTime>T#0s);
EachCtlPrd_ATHeat_inst(In:= ATHeatPhase & NOT(EachCtlPrd_ATHeat_inst.Q),
  PT:= CtlPrd_Heat);

ATCoolPhase := PIDAT_HeatCool_inst.ATBusy & (MVCoolTime>T#0s);
EachCtlPrd_ATCool_inst(In:= ATCoolPhase & NOT(EachCtlPrd_ATCool_inst.Q),
  PT:= CtlPrd_Cool);

// Выдача сигнала нагрева на модуль CJ1W-OC201
TPOHeat_inst(Enable :=NOT(PIDAT_HeatCool_inst.ATBusy),
  AIn :=MV_Heat,
  CtlPrd :=CtlPrd_Heat );
AT_Heat_inst(In:= ATHeatPhase & (MVHeatTime<>CtlPrd_Heat) & NOT(EachCtlPrd_ATHeat_i
nst.Q) ,
  PT:= MVHeatTime);
J02_Ch1_Out00 :=( TPOHeat_inst.DOut ) OR
  ( ATHeatPhase & (MVHeatTime=CtlPrd_Heat)) OR
  ( AT_Heat_inst.Q & ATHeatPhase );

// Выдача сигнала охлаждения на CJ1W-OC201
TPOCool_inst(Enable :=NOT(PIDAT_HeatCool_inst.ATBusy),
  AIn :=MV_Cool,
  CtlPrd :=CtlPrd_Cool );
AT_Cool_inst(In:= ATCoolPhase & (MVCoolTime<>CtlPrd_Cool) & NOT(EachCtlPrd_ATCool_i
nst.Q) ,
  PT:= MVCoolTime);
J02_Ch1_Out04 :=( TPOCool_inst.DOut ) OR
  ( ATCoolPhase & (MVCoolTime=CtlPrd_Cool)) OR
  ( AT_Cool_inst.Q & ATCoolPhase );

// Получение значений для вывода на сенсорную панель.
PTOut_PV := PV;

PTOut_ATDone := PIDAT_HeatCool_inst.ATDone;
PTOut_ATBusy := PIDAT_HeatCool_inst.ATBusy;
PTOut_Error := PIDAT_HeatCool_inst.Error;

PTOut_MV := PIDAT_HeatCool_inst.MV;
PTOut_MVHeat := PIDAT_HeatCool_inst.MV_Heat;
PTOut_MVCool := PIDAT_HeatCool_inst.MV_Cool;

PT_PB_Heat := PB_Heat;
PT_TI_Heat := TimeToNanoSec(TI_Heat)/1000000;
PT_TD_Heat := TimeToNanoSec(TD_Heat)/1000000;

PT_PB_Cool := PB_Cool;
PT_TI_Cool := TimeToNanoSec(TI_Cool)/1000000;
PT_TD_Cool := TimeToNanoSec(TD_Cool)/1000000;
```

```
PT_ManMV := ManMV;
```

TimeProportionalOut

Команда TimeProportionalOut преобразует управляющее воздействие в выходной сигнал для широтно-импульсного регулирования (ШИР).

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TimeProportionalOut	Выход широтно-импульсного регулирования	FB		TimeProportionalOut_instance(Enable, AIn, CtlPrd, MinPlsWidth, Delay, DOut, Error);

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Enable	Активация	Вход	ИСТИНА: выполнить ЛОЖЬ: сбросить выход ШИР	Зависит от типа данных.	---	ЛОЖЬ
AIn	Управляющее воздействие		Управляющее воздействие	0...100	%	0
CtlPrd	Период регулирования		Период следования импульсов ШИР	T#0.1 с ... T#100 с	с	T#2 с
MinPlsWidth	Минимальная ширина импульса		Минимальная ширина импульса	0...50	%	1
Delay	Задержка		Время задержки включения	0...100	%	0
DOut	Выход широтно-импульсного регулирования	Выход	ИСТИНА: выход широтно-импульсного регулирования включен. ЛОЖЬ: выход широтно-импульсного регулирования выключен.	Зависит от типа данных.	---	---

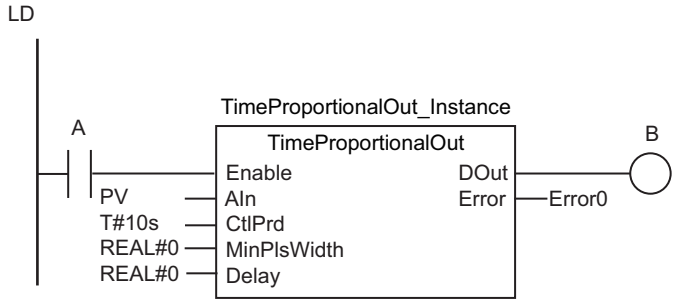
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	ОК																			
AIn														ОК						
CtlPrd																ОК				
MinPlsWidth														ОК						
Delay														ОК						
DOut	ОК																			

Функция

Команда TimeProportionalOut преобразует сигнал управляющего воздействия, например полученный в результате ПИД-регулирования, в выходной сигнал для широтно-импульсного регулирования (ШИР).
Фактически значение управляющего воздействия преобразуется в эквивалентное соотношение между временем включенного состояния и временем выключенного состояния выхода ШИР (иначе говоря, управляющее воздействие преобразуется в эквивалентный коэффициент заполнения импульсов).

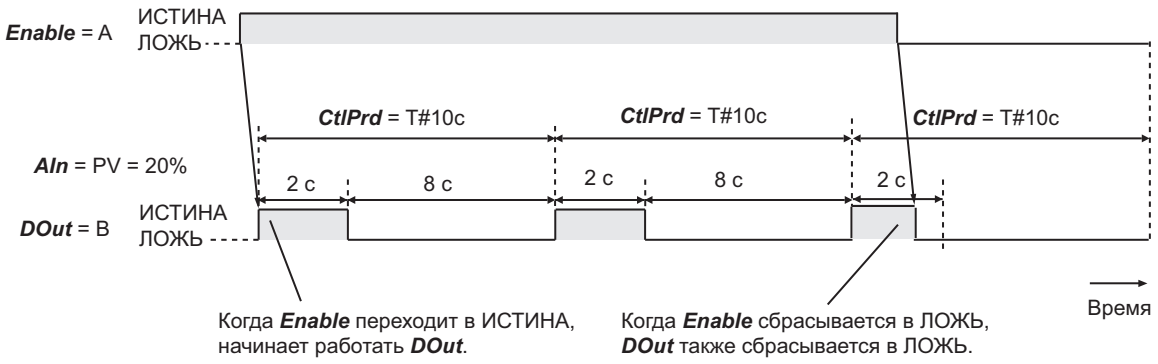
Пока значение *Enable* = ИСТИНА, значение управляющего воздействия *AIn* преобразуется в сигнал для широтно-импульсного регулирования *DOut* с заданным периодом регулирования (т. е. периодом импульсов) *CtlPrd*.
Если значение *Enable* поменяется на ЛОЖЬ, выход широтно-импульсного регулирования будет сброшен.
Выходы *DOut* и *Error* перейдут в состояние ЛОЖЬ. Значения *CtlPrd*, *MinPlsWidth* и *Delay* обновляются, когда вход *Enable* переходит из состояния ЛОЖЬ в состояние ИСТИНА.

В приведенном ниже примере значение *CtlPrd* = 10 с, а значение *AIn* = 20 %. Пока *Enable* = ИСТИНА, *DOut* = ИСТИНА в течение двух секунд, а затем ЛОЖЬ в течение восьми секунд. Это повторяется с периодом в 10 с.



ST

```
TimeProportionalOut_instance(A,PV,T#10s,REAL#0,REAL#0,B,Error0);
```



Разрешение выхода широтно-импульсного регулирования *DOut*

Под разрешением выхода *DOut* понимается минимальное приращение значения *AIn*, которое может быть распознано и преобразовано в соответствующее приращение значения *DOut*. Если разрешающая способность сигнала *AIn* выше разрешающей способности *DOut*, значения *AIn* при преобразовании в значения *DOut* округляются в соответствии с разрешающей способностью *DOut*.

Разрешение выхода *DOut* определяется по следующей формуле:

Разрешение *DOut* (%) = период выполнения задачи / *CtlPrd* x 100

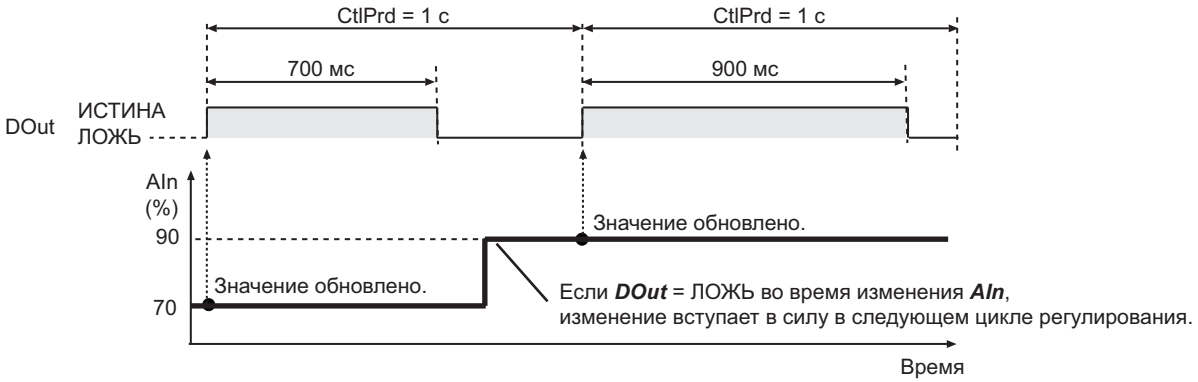
Например, если период выполнения задачи = 1 мс, а значение *CtlPrd* = 1 с, то разрешение *DOut* = 0,1%. В этом случае значение *AIn* округляется до одного знака после запятой.

Время обновления значения управляющего воздействия *AIn*

Момент обновления значения *AIn* зависит от того, чему равно значение *DOut*: ЛОЖЬ или ИСТИНА.

● *DOut* = ЛОЖЬ

Если *DOut* = ЛОЖЬ, любое изменение значения *AIn* применяется в следующем цикле регулирования.

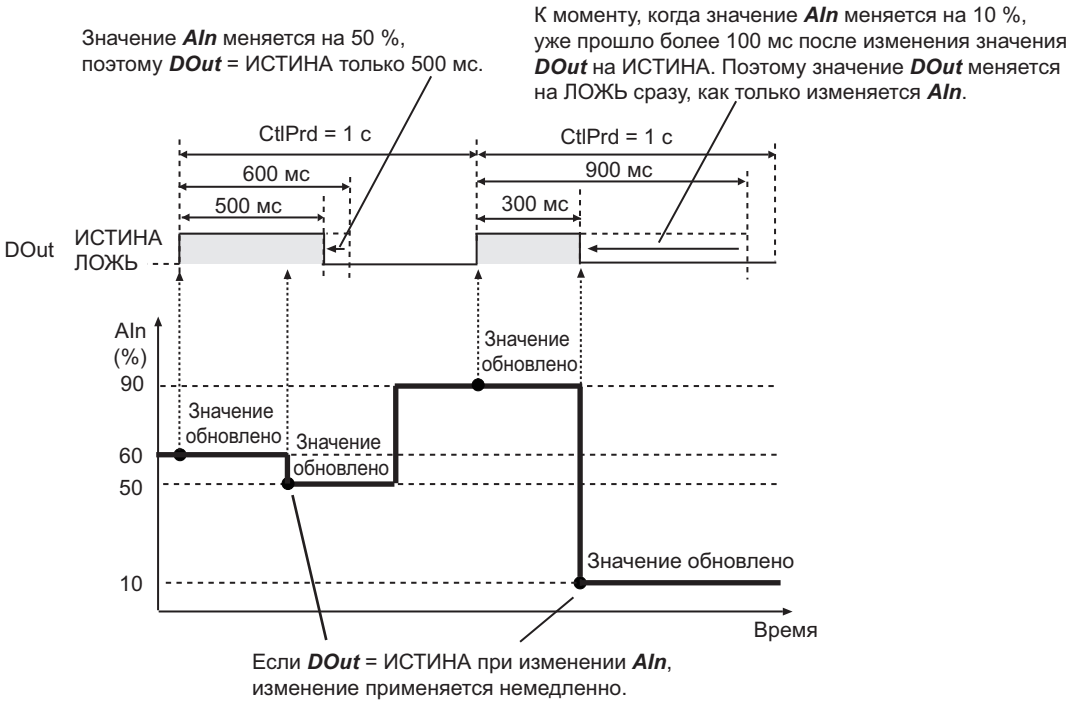


● *DOut* = ИСТИНА

Если *DOut* = ИСТИНА, любое изменение значения *AIn* вступает в силу немедленно.

В примере на рисунке ниже период следования импульсов (период регулирования) *CtlPrd* составляет 1 с.

- Предположим, что на начало периода регулирования значение *AIn* было равно 60 %. Если значение *AIn* поменяется на 50 %, когда *DOut* = ИСТИНА, *DOut* будет равно ИСТИНА только в течение 500 мс.
- Предположим теперь, что значение *AIn* было равно 90 % на начало периода регулирования и что значение *AIn* стало равно 10 % через 300 мс после того, как значение *DOut* поменялось на ИСТИНА. В этом случае 100 мс, которые эквивалентны значению 10 % от периода регулирования, уже прошли, поэтому значение *DOut* меняется на ЛОЖЬ немедленно.



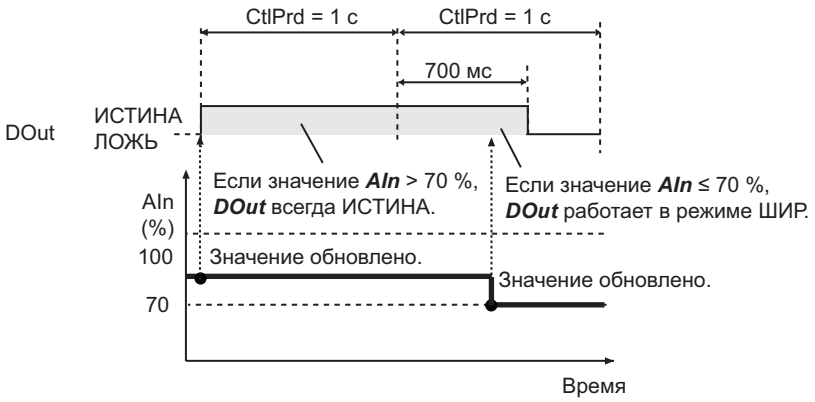
Работа выхода широтно-импульсного регулирования *DOut* в зависимости от минимальной ширины импульса *MinPlsWidth*

Минимальная ширина импульса — это минимальное время, в течение которого выход *DOut* находится в состоянии ИСТИНА или в состоянии ЛОЖЬ.
Задав минимальную длительность импульса с помощью параметра *MinPlsWidth*, можно уменьшить «дребезг» сигнала *DOut*.
Это, например, позволит уменьшить частоту включения и выключения вентилятора при регулировании охлаждения и тем самым уменьшить энергопотребление.

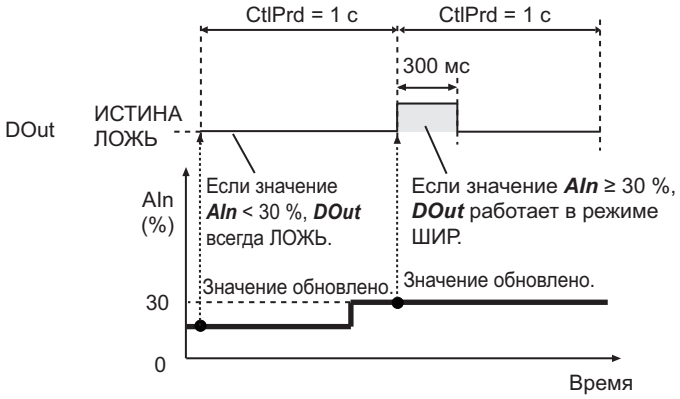
В следующей таблице показана работа выхода *DOut* при разных соотношениях значений *MinPlsWidth* и *AIn*.

Соотношение между значениями *MinPlsWidth* и *AIn*	Работа выхода *DOut*
AIn < *MinPlsWidth*	Всегда ЛОЖЬ
MinPlsWidth ≤ *AIn* ≤ 100 - *MinPlsWidth*	Выход широтно-импульсного регулирования
AIn > 100 - *MinPlsWidth*	Всегда ИСТИНА

В примере на рисунке ниже показана работа выхода *DOut* для случая, когда *MinPlsWidth* = 30 %. Если значение *AIn* больше 70 %, выход всегда находится в состоянии ИСТИНА. При значении 70 % или меньше выход работает в режиме широтно-импульсного регулирования.



Если значение *AIn* меньше 30 %, выход всегда пребывает в состоянии ЛОЖЬ. При значении 30 % или больше выход работает в режиме широтно-импульсного регулирования.



Работа выхода широтно-импульсного регулирования *DOut* в зависимости от задержки *Delay*

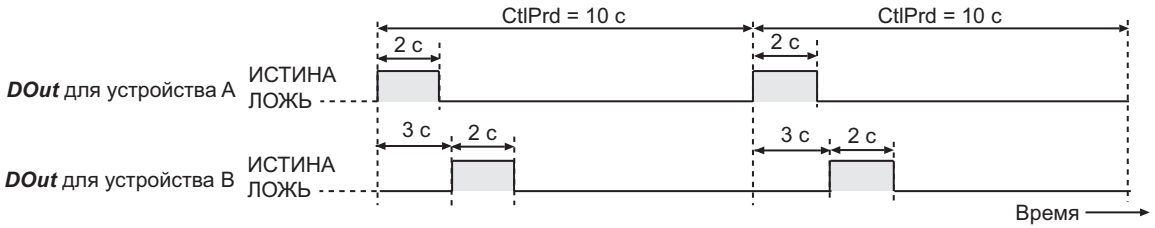
Выход *DOut* не переходит в состояние ИСТИНА до тех пор, пока не истекает заданное время задержки, отсчитываемое от начала периода регулирования.
Если в программе выполняется несколько экземпляров команд TimePropotionalOut, с помощью параметра *Delay* можно задать разное время перехода выхода *DOut* в состояние ИСТИНА для каждой команды.
Это уменьшит вероятность того, что выход *DOut* будет включен одновременно несколькими командами.
Например, если в системе используется несколько нагревательных устройств, для каждого устройства можно индивидуально задать задержку *Delay*, чтобы исключить одновременную работу двух нагревательных устройств и, соответственно, большой ток потребления от сети.

Выход *DOut* переходит в состояние ИСТИНА по истечении времени, заданного параметром *Delay* (в процентах), с начала периода регулирования.

Например, можно задать следующие значения для устройств A и B, имеющих одинаковый период регулирования.

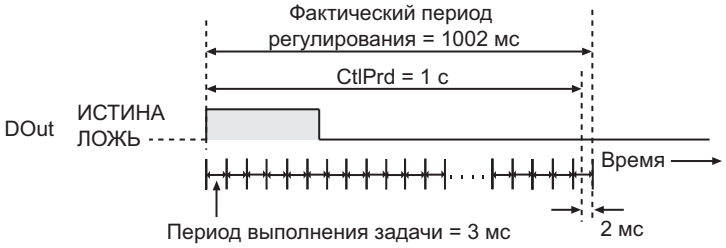
Устройство	Значение *Delay*	Значение *AIn*	Значение *CtlPrd*
Устройство A	0 %	20 %	10 с
Устройство B	30 %		

DOut для устройства A переходит в состояние ИСТИНА в начале периода регулирования. *DOut* для устройства B переходит в состояние ИСТИНА через 3 секунды после начала периода регулирования.



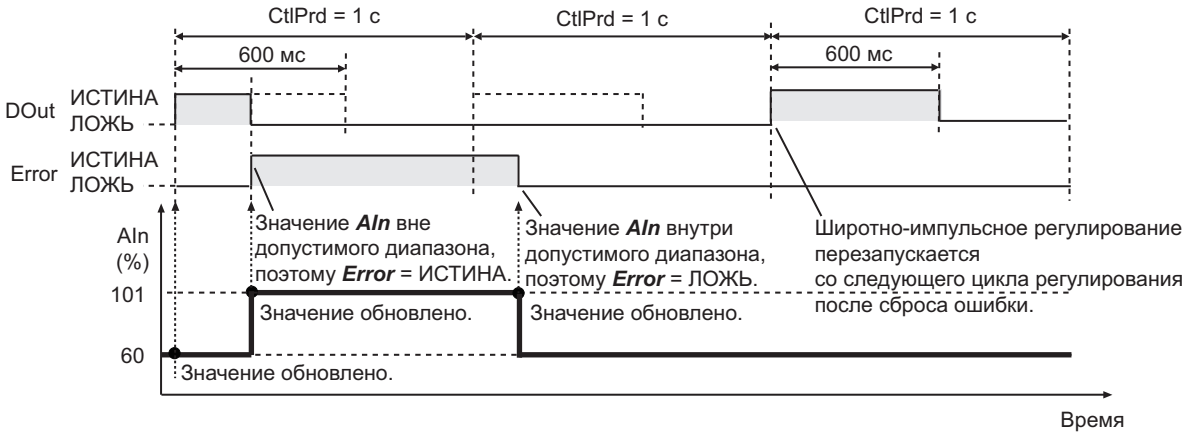
Меры предосторожности для обеспечения надлежащей эксплуатации

- Значение периода регулирования, задаваемое в параметре *CtlPrd*, должно быть кратно периоду выполнения задачи, которой назначена программа.
  Если период выполнения задачи не кратен значению *CtlPrd*, цикл регулирования не завершается с истечением заданного периода регулирования, а длится до следующего цикла выполнения задачи, т. е. фактический период регулирования несколько отличается от заданного. Например, если период выполнения задачи установлен равным 3 мс, а значение *CtlPrd* = 1 с, фактический период регулирования будет равен 1002 мс (добавляется 2 мс — время с момента завершения *CtlPrd* до следующего выполнения задачи).

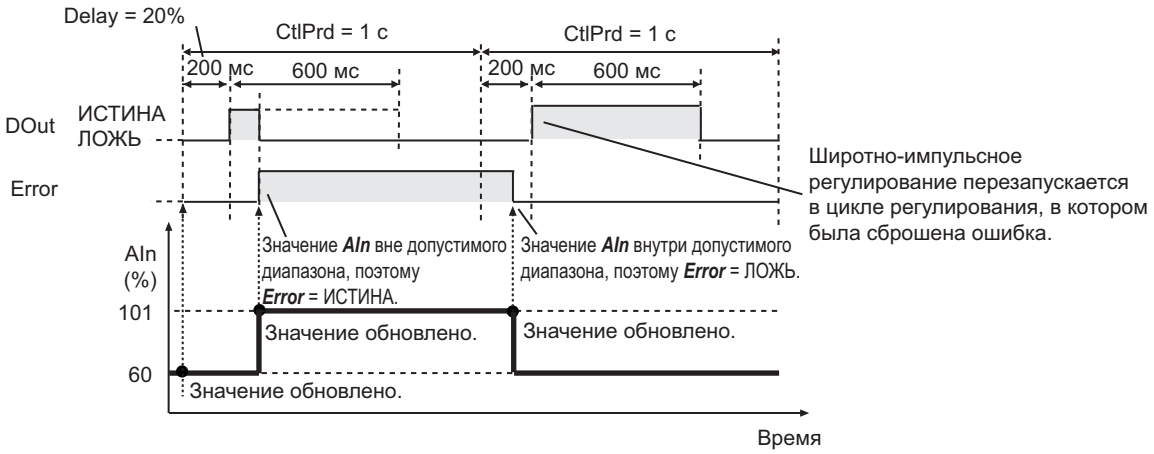


- Период выполнения задачи и период регулирования *CtlPrd* следует задавать с таким расчетом, чтобы разрешение выхода *DOut* составляло 0,1 % или меньше. Если разрешение *DOut* будет превышать 0,1 %, несоответствие между коэффициентом заполнения импульсов *DOut* и значением *AIn* будет слишком большим и качество регулирования будет недостаточно высоким. Например, если *CtlPrd* = 10 с, период выполнения задачи следует установить равным 10 мс или меньше.
- Если используется несколько экземпляров этой команды и нужно синхронизировать их периоды регулирования, команды следует использовать в одной программе. Если команды будут использоваться в разных программах, их периоды регулирования будут зависеть от времени выполнения программ и не будут синхронизированы между собой.
- Временной интервал между моментом перехода входа *Enable* в состояние ИСТИНА и началом работы выхода *DOut* не является постоянным.
- Выход значения *AIn*, *CtlPrd*, *MinPlsWidth* или *Delay* за пределы допустимого диапазона приводит к возникновению ошибки. Значение *Error* меняется на ИСТИНА, а значение *DOut* меняется на ЛОЖЬ. Если значение *AIn* выйдет за пределы допустимого диапазона, дальнейшая работа выхода *DOut* зависит от того, когда будет сброшена ошибка. Сказанное поясняется на рисунках ниже.

a) Если ошибка сбрасывается после того, как выход *DOut* должен был перейти в состояние ИСТИНА, работа выхода *DOut* в режиме ШИР возобновляется со следующего цикла регулирования.

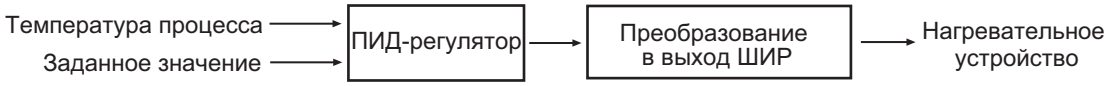


b) Если же ошибка сбрасывается до наступления момента перехода выхода *DOut* в состояние ИСТИНА, выход *DOut* начинает работать в режиме ШИР в том же цикле регулирования, в котором произведен сброс ошибки.



Пример программы

Рассмотрим пример программы, которая осуществляет регулирование температуры в четырех точках, подает аварийные сигналы при выходе температуры за верхнюю или нижнюю предельные границы, а также сигнализирует чрезмерное отклонение температуры в сторону увеличения или в сторону уменьшения. Применяется ПИД-регулирование. Получаемые в результате ПИД-регулирования управляющие воздействия преобразуются в импульсные сигналы для широтно-импульсного регулирования, которые подаются на нагревательные устройства.



Характеристики

В таблице ниже указано используемое оборудование и параметры регулирования температуры.

Устройство/параметр	Характеристики
Модуль входов	CJ1W-PH41U (модуль универсальных входов с гальванической развязкой)
Типы входов	Термопары типа K
Модуль выходов	CJ1W-OD212 (модуль транзисторных выходов)
Заданное значение	100 °C
Верхняя предельная температура	200 °C
Нижняя предельная температура	0 °C
Гистерезис аварии по выходу за верхний/нижний предел	5 °C
Верхнее отклонение температуры	50 °C
Нижнее отклонение температуры	50 °C
Гистерезис аварии по верхнему/нижнему отклонению	3 °C
Период измерения для ПИД-регулирования	100 мс
Период выходного сигнала управления	1 с

Конфигурация и настройки

В таблице ниже приведены используемые параметры модуля входов CJ1W-PH41U.

Параметр	Заданное значение
Input1: тип входного сигнала	K(1)
Input2: тип входного сигнала	K(1)
Input3: тип входного сигнала	K(1)
Input4: тип входного сигнала	K(1)

В таблице ниже перечислены используемые параметры карты соответствия входов и выходов (I/O map).

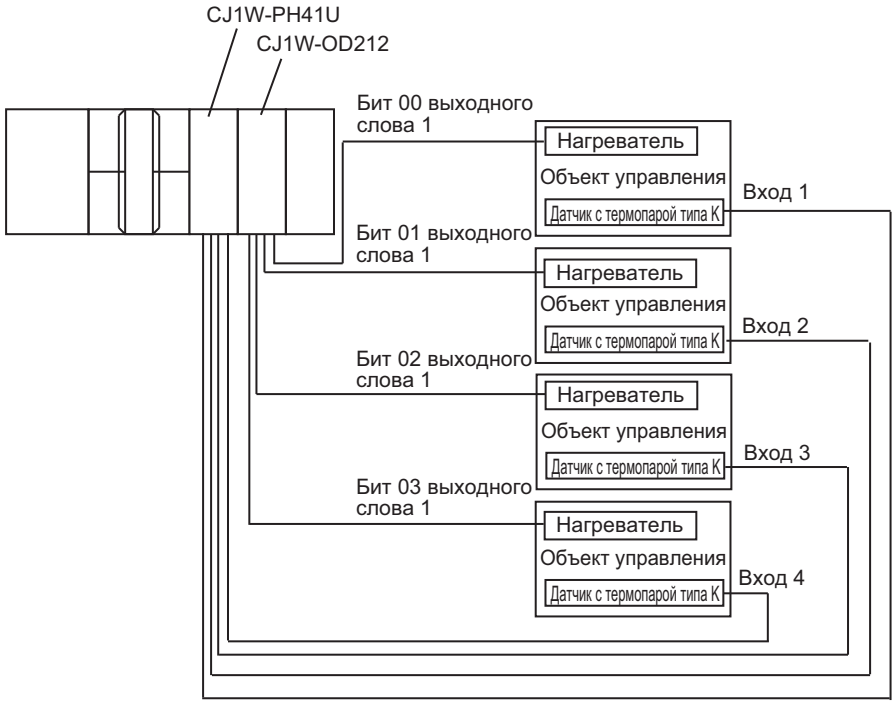
Модуль	Порт ввода-вывода	Описание	Переменная
CJ1W-PH41U	Ch1_AIInPV	Вход 1, регулируемая величина (значение типа INT)	AI1
	Ch2_AIInPV	Вход 2, регулируемая величина (значение типа INT)	AI2
	Ch3_AIInPV	Вход 3, регулируемая величина (значение типа INT)	AI3
	Ch4_AIInPV	Вход 4, регулируемая величина (значение типа INT)	AI4
CJ1W-OD212	Ch1_Out00	Бит 00 выходного слова 1	DO1
	Ch1_Out01	Бит 01 выходного слова 1	DO2
	Ch1_Out02	Бит 02 выходного слова 1	DO3
	Ch1_Out03	Бит 03 выходного слова 1	DO4

Ниже перечислены входы и выходы, которые используются для регулирования температуры в четырех точках.

Вход	Выход
AI1	DO1
AI2	DO2
AI3	DO3
AI4	DO4

Период выполнения задачи, которой назначена программа, равен 1 мс.

● **Структурная схема**



Порядок выполнения операций

Для каждой из четырех точек измерения соблюдается следующий порядок операций.

1 Считывается температура процесса.

2 Используется команда LimitAlarm_REAL для выдачи сигналов аварии по выходу за верхний/нижний предел для температуры процесса.

3 В случае, если возникла ошибка в команде LimitAlarm_REAL или подан сигнал аварии по выходу за верхний или нижний предел, выдается сигнал для обеспечения безопасности.

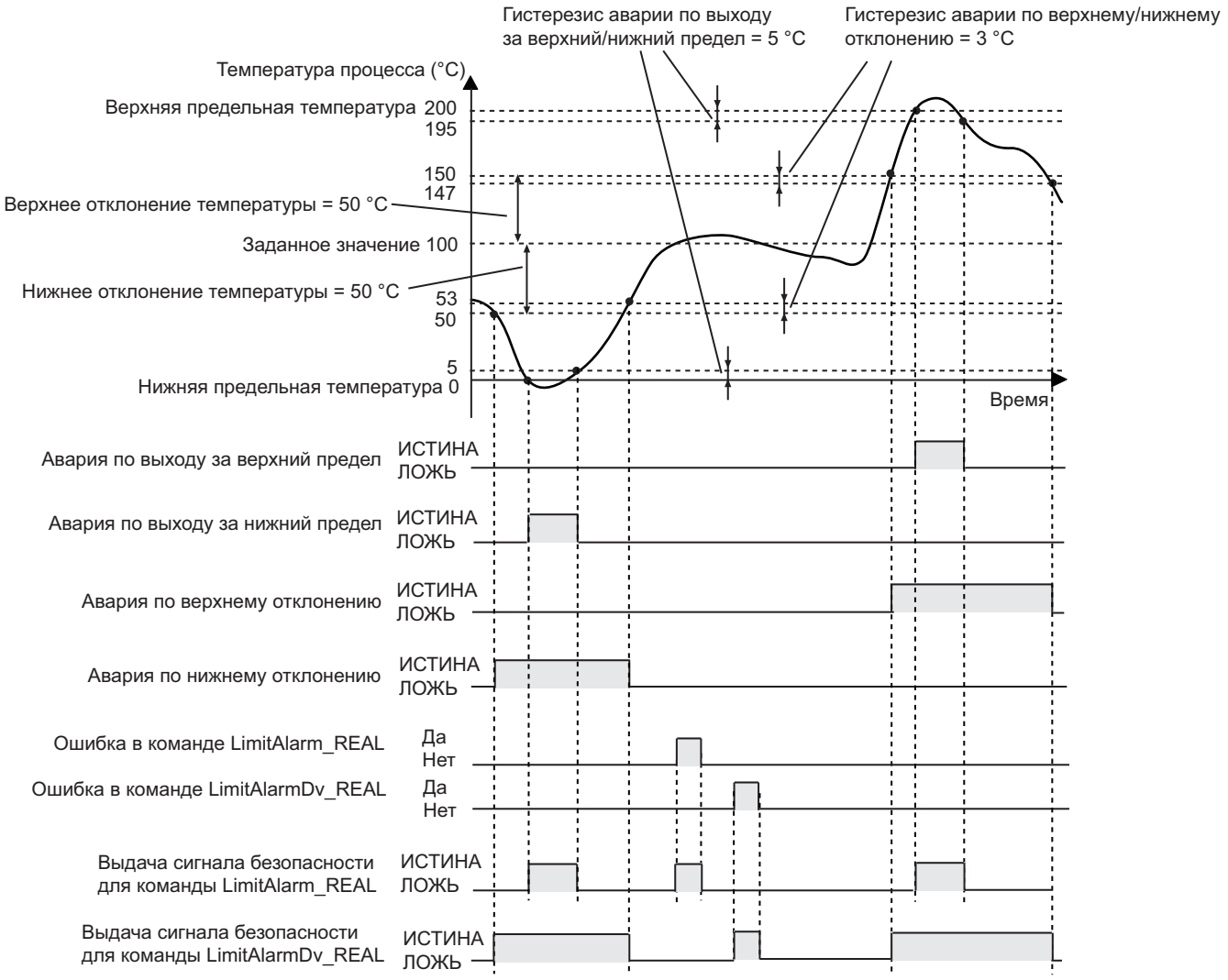
4 Используется команда LimitAlarmDv_REAL для выдачи сигналов аварии по верхнему/нижнему отклонению температуры процесса от заданного значения.

5 В случае, если возникла ошибка в команде LimitAlarmDv_REAL или подан сигнал аварии по верхнему/нижнему отклонению, выдается сигнал для обеспечения безопасности.

6 Выполняется регулирование температуры с использованием команды PIDAT.

7 Используется команда TimeProportionalOut для преобразования управляющего воздействия с выхода ПИД-регулятора в сигнал широтно-импульсного регулирования, подаваемый на нагревательное устройство.

● Формирование сигналов аварии по выходу за верхний/нижний предел и аварии по верхнему/нижнему отклонению



Определения глобальных переменных

● Глобальные переменные

Переменная	Тип данных	Параметр «AT»*1	Комментарий
AI1	INT	IOBus://rack#0/slot#0/Ch1_AIInPV	Вход 1, регулируемая величина (значение типа INT)
AI2	INT	IOBus://rack#0/slot#0/Ch2_AIInPV	Вход 2, регулируемая величина (значение типа INT)
AI3	INT	IOBus://rack#0/slot#0/Ch3_AIInPV	Вход 3, регулируемая величина (значение типа INT)
AI4	INT	IOBus://rack#0/slot#0/Ch4_AIInPV	Вход 4, регулируемая величина (значение типа INT)
DO1	BOOL	IOBus://rack#0/slot#1/Ch1_Out/Ch1_Out00	Бит 00 выходного слова 1
DO2	BOOL	IOBus://rack#0/slot#1/Ch1_Out/Ch1_Out01	Бит 01 выходного слова 1

Переменная	Тип данных	Параметр «AT»*1	Комментарий
DO3	BOOL	IOBus://rack#0/slot#1/ Ch1_Out/Ch1_Out02	Бит 02 выходного слова 1
DO4	BOOL	IOBus://rack#0/slot#1/ Ch1_Out/Ch1_Out03	Бит 03 выходного слова 1

*1. В таблице приведены переменные для модуля входов CJ1W-PH41U, установленного в слот №0 стойки №0, и для модуля выходов CJ1W-OD212, установленного в слот №1 той же стойки.

Примечание Глобальные переменные для порта каждого модуля генерируются автоматически на основе настроенного соответствия входов-выходов.

Программа на языке LD

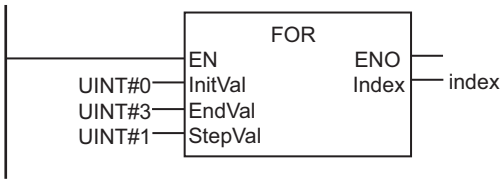
Внутренние переменные	Имя	Тип данных	По умолчанию	Сохранение	Комментарий
	index	UINT	0	☐	Счетчик цикла
	LimitAlarm_ON	BOOL	Истина	☐	Выполнение команды аварии выхода за верхний/нижний предел
	LimitAlarmDv_ON	BOOL	Истина	☐	Выполнение команды аварии верхнего/нижнего отклонения
	TimeProportionalOut_ON	BOOL	Истина	☐	Выполнение команды TimeproportionalOutput
	AI	INT	0	☐	Текущее значение
	PV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Регулируемая величина
	SP	ARRAY[0..3] OF REAL	[4(100)]	☐	Заданное значение
	DOut_TPO	BOOL	Ложь	☐	Выход широтно-импульсного регулирования
	HighVal	ARRAY[0..3] OF REAL	[4(200)]	☐	Установленный верхний предел для сигнализации аварии по выходу за верхний/нижний предел
	LowVal	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Установленный нижний предел для сигнализации аварии по выходу за верхний/нижний предел
	Hystrs_LimitAlarm	ARRAY[0..3] OF REAL	[4(5)]	☐	Гистерезис аварии по выходу за верхний/нижний предел

Внутренние переменные	Имя	Тип данных	По умолчанию	Сохранение	Комментарий
	Q_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выход сигнала аварии по выходу за верхний/нижний предел
	HighAlm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по выходу за верхний предел
	LowAlm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по выходу за нижний предел
	Error_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде LimitAlarm_REAL
	Alm_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал для обеспечения безопасности для команды аварии выхода за верхний/нижний предел
	DvHighVal	ARRAY[0..3] OF REAL	[4(50)]	☐	Установленное верхнее отклонение для сигнализации аварии по верхнему/нижнему отклонению
	DvLowVal	ARRAY[0..3] OF REAL	[4(50)]	☐	Установленное нижнее отклонение для сигнализации аварии по верхнему/нижнему отклонению
	Q_LimitAlarmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выход сигнала аварии по верхнему/нижнему отклонению
	HighAlmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по верхнему отклонению
	LowAlmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по нижнему отклонению
	Error_LimitAlarmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде LimitAlarmDv_REAL
	Hystrs_LimitAlarmDv	ARRAY[0..3] OF REAL	[4(3)]	☐	Гистерезис аварии по верхнему/нижнему отклонению
	Alm_LimitAlarmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал для обеспечения безопасности для команды аварии по верхнему/нижнему отклонению
	Run	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Условие выполнения
	ManCtl	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ручное/автоматическое управление
	StartAT	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Условие выполнения автонастройки

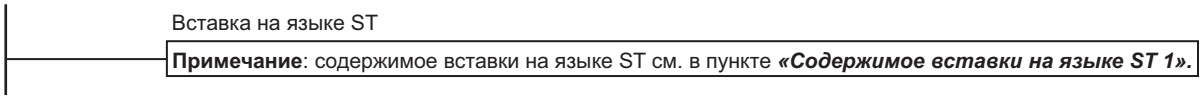
Внутренние переменные	Имя	Тип данных	По умолчанию	Сохранение	Комментарий
	OprSetParams	_sOPR_SET_PARAMS	(MVLowLmt:=0.0, MVUpLmt:=100.0, ManResetVal:=0.0, MVTrackSw:=False, MVTrackVal:=0.0, StopMV:=0.0, ErrorMV:=0.0,Alpha:=0.65, ATCalcGain:=1.0, ATHystrs:=0.2)	☐	Рабочие параметры
	InitSetParams	_sINIT_SET_PARAMS	(SampTime:=T#100 ms, RngLowLmt:=-10.0, RngUpLmt:=1000.0, DirOpr:=False)	☐	Начальные параметры
	PB	ARRAY[0..3] OF REAL	[4(10)]	☑	Зона пропорциональности
	TI	ARRAY[0..3] OF TIME	[4(T#0 c)]	☑	Постоянная времени интегрирования
	TD	ARRAY[0..3] OF TIME	[4(T#0 c)]	☑	Постоянная времени дифференцирования
	ManMV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Ручное управляющее воздействие
	ATDone	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Нормальное завершение автонастройки
	ATBusy	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выполнение автонастройки
	Error_PIDAT	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде PIDAT
	ErrorID	ARRAY[0..3] OF WORD	[4(16#0)]	☐	Идентификатор ошибки для команды PIDAT
	MV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Управляющее воздействие
	CtlPrd	ARRAY[0..3] OF TIME	[4(T#1 c)]	☐	Период регулирования
	MinPlsWidth	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Минимальная ширина импульса
	Delay	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Время задержки включения
	Error_TimeProportionalOut	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде TimeProportionalOut
	LimitAlarm_REAL_instance	ARRAY[0..3] OF LimitAlarm_REAL		☐	
	LimitAlarmDv_REAL_instance	ARRAY[0..3] OF LimitAlarmDv_REAL		☐	
	PIDAT_instance	ARRAY[0..3] OF PIDAT		☐	
	TimeProportionalOut_instance	ARRAY[0..3] OF TimeProportionalOut		☐	

Внешние переменные	Имя	Тип данных	Комментарий
	AI1	INT	Вход №1 (регулируемая величина)
	AI2	INT	Вход №2 (регулируемая величина)
	AI3	INT	Вход №3 (регулируемая величина)
	AI4	INT	Вход №4 (регулируемая величина)
	DO1	BOOL	Выходное слово 1 (бит 00)
	DO2	BOOL	Выходное слово 1 (бит 01)
	DO3	BOOL	Выходное слово 1 (бит 02)
	DO4	BOOL	Выходное слово 1 (бит 03)

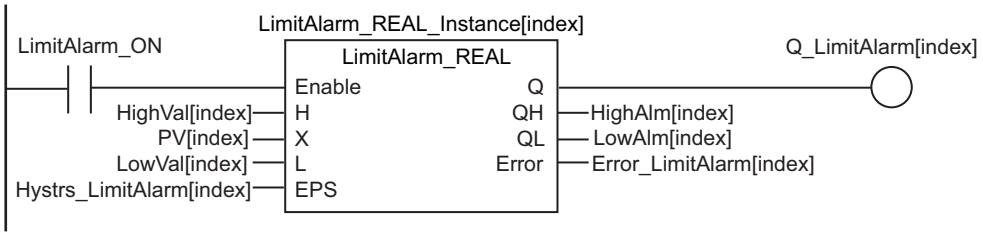
Регулирование температуры в четырех точках.



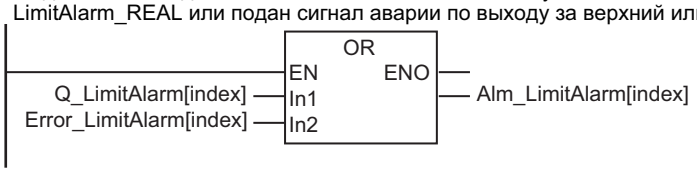
Получение значения регулируемой величины.



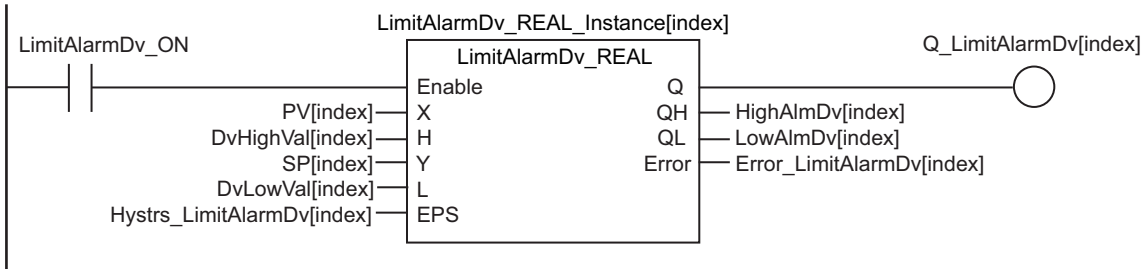
Сигнализация аварии по выходу за верхний/нижний предел



Выдача сигнала для обеспечения безопасности в случае, если возникла ошибка в команде LimitAlarm_REAL или подан сигнал аварии по выходу за верхний или нижний предел.

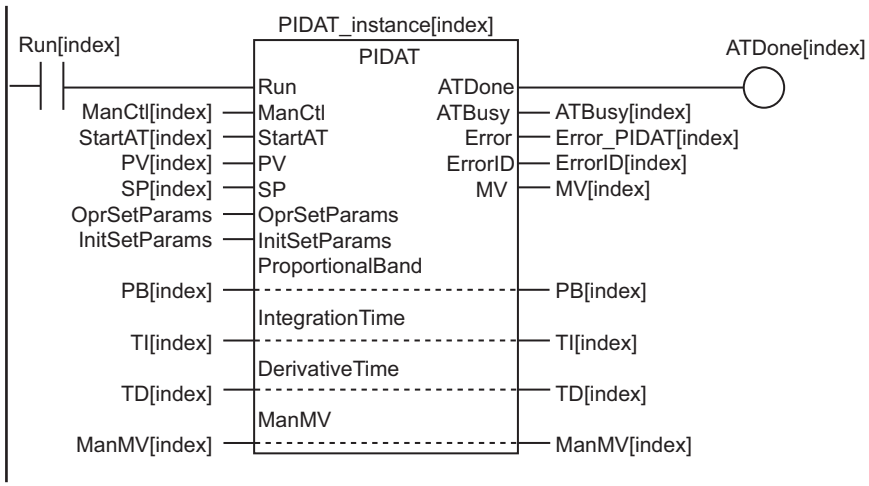


Сигнализация аварии по верхнему/нижнему отклонению

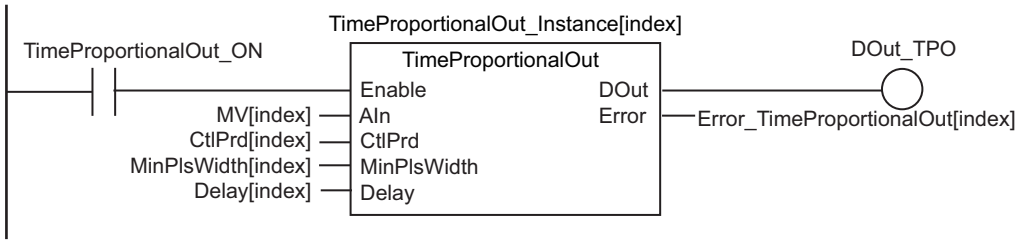


Выдача сигнала для обеспечения безопасности в случае, если возникла ошибка в команде LimitAlarmDv_REAL или подан сигнал аварии по выходу за верхний или нижний предел.

Выполнение команды PIDAT.



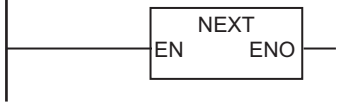
Выход широтно-импульсного регулирования



Вывод значений в биты 00…03 выходного слова 1.

Вставка на языке ST

Примечание: содержимое вставки на языке ST см. в пункте ***«Содержимое вставки на языке ST 2».***



● Содержимое вставки на языке ST 1

```
//Получение входных значений 1...4.
CASE index OF
INT#0:
  AI:=AI1;
INT#1:
  AI:=AI2;
INT#2:
  AI:=AI3;
ELSE
  AI:=AI4;
END_CASE;

//Преобразование AI (рег. велич.) в вещественное число.
PV[index]:=INT_TO_REAL(AI)/REAL#10.0;  // CJ1W-PH41U выдает значение, которое в 10
раз больше фактического, поэтому его нужно поделить на 10,0.
```

● Содержимое вставки на языке ST 2

```
//Вывод значений в биты 00…03 выходного слова 1.
CASE index OF
INT#0:
  DO1:=DOut_TPO;
INT#1:
  DO2:=DOut_TPO;
INT#2:
  DO3:=DOut_TPO;
ELSE
  DO4:=DOut_TPO;
END_CASE;
```

Программа на языке ST

Внутренние переменные	Имя	Тип данных	По умолчанию	Сохранение	Комментарий
	index	UINT	0	☐	Счетчик цикла
	LimitAlarm_ON	BOOL	Истина	☐	Выполнение команды аварии выхода за верхний/нижний предел
	LimitAlarmDv_ON	BOOL	Истина	☐	Выполнение команды аварии верхнего/нижнего отклонения
	TimeProportionalOut_ON	BOOL	Истина	☐	Выполнение команды широтно-импульсного регулирования
	AI	INT	0	☐	Текущее значение

Внутренние переменные	Имя	Тип данных	По умолчанию	Сохранение	Комментарий
	PV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Регулируемая величина
	SP	ARRAY[0..3] OF REAL	[4(100)]	☐	Заданное значение
	DOut_TPO	BOOL	Ложь	☐	Выход широтно-импульсного регулирования
	HighVal	ARRAY[0..3] OF REAL	[4(200)]	☐	Установленный верхний предел для сигнализации аварии по выходу за верхний/нижний предел
	LowVal	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Установленный нижний предел для сигнализации аварии по выходу за верхний/нижний предел
	Hystrs_LimitAlarm	ARRAY[0..3] OF REAL	[4(5)]	☐	Гистерезис аварии по выходу за верхний/нижний предел
	Q_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выход сигнала аварии по выходу за верхний/нижний предел
	HighAlm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по выходу за верхний предел
	LowAlm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по выходу за нижний предел
	Error_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде LimitAlarm_REAL
	Alm_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал для обеспечения безопасности для команды аварии выхода за верхний/нижний предел
	DvHighVal	ARRAY[0..3] OF REAL	[4(50)]	☐	Установленное верхнее отклонение для сигнализации аварии по верхнему/нижнему отклонению
	DvLowVal	ARRAY[0..3] OF REAL	[4(50)]	☐	Установленное нижнее отклонение для сигнализации аварии по верхнему/нижнему отклонению
	Q_LimitAlarmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выход сигнала аварии по верхнему/нижнему отклонению
	HighAlmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по верхнему отклонению

Внутренние переменные	Имя	Тип данных	По умолчанию	Сохранение	Комментарий
	LowAlmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по нижнему отклонению
	Error_LimitAlarmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде LimitAlarmDv_REAL
	Hystrs_LimitAlarmDv	ARRAY[0..3] OF REAL	[4(3)]	☐	Гистерезис аварии по верхнему/нижнему отклонению
	Alm_LimitAlarmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал для обеспечения безопасности для команды аварии по верхнему/нижнему отклонению
	Run	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Условие выполнения
	ManCtl	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ручное/автоматическое управление
	StartAT	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Условие выполнения автонастройки
	OprSetParams	_sOPR_SET_PARAMS	(MVLowLmt:=0.0, MVUpLmt:=100.0, ManResetVal:=0.0, MVTrackSw:=False, MVTrackVal:=0.0, StopMV:=0.0, ErrorMV:=0.0,Alpha:=0.65, ATCalcGain:=1.0, ATHystrs:=0.2)	☐	Рабочие параметры
	InitSetParams	_sINIT_SET_PARAMS	(SampTime:=T#100 ms, RngLowLmt:=-10.0,RngUpLmt:=1000.0, DirOpr:=False)	☐	Начальные параметры
	PB	ARRAY[0..3] OF REAL	[4(10)]	☑	Зона пропорциональности
	TI	ARRAY[0..3] OF TIME	[4(T#0 c)]	☑	Постоянная времени интегрирования
	TD	ARRAY[0..3] OF TIME	[4(T#0 c)]	☑	Постоянная времени дифференцирования
	ManMV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Ручное управляющее воздействие
	ATDone	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Нормальное завершение автонастройки
	ATBusy	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выполнение автонастройки
	Error_PIDAT	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде PIDAT
	ErrorID	ARRAY[0..3] OF WORD	[4(16#0)]	☐	Идентификатор ошибки для команды PIDAT
	MV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Управляющее воздействие
	CtlPrd	ARRAY[0..3] OF TIME	[4(T#1 c)]	☐	Период регулирования

Внутренние переменные	Имя	Тип данных	По умолчанию	Сохранение	Комментарий
	MinPlsWidth	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Минимальная ширина импульса
	Delay	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Время задержки включения
	Error_TimeProportionalOut	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде TimeProportionalOut
	LimitAlarm_REAL_instance	ARRAY[0..3] OF LimitAlarm_REAL		☐	
	LimitAlarmDv_REAL_instance	ARRAY[0..3] OF LimitAlarmDv_REAL		☐	
	PIDAT_instance	ARRAY[0..3] OF PIDAT		☐	
	TimeProportionalOut_instance	ARRAY[0..3] OF TimeProportionalOut		☐	

Внешние переменные	Имя	Тип данных	Комментарий
	AI1	INT	Вход №1 (регулируемая величина)
	AI2	INT	Вход №2 (регулируемая величина)
	AI3	INT	Вход №3 (регулируемая величина)
	AI4	INT	Вход №4 (регулируемая величина)
	DO1	BOOL	Выходное слово 1 (бит 00)
	DO2	BOOL	Выходное слово 1 (бит 01)
	DO3	BOOL	Выходное слово 1 (бит 02)
	DO4	BOOL	Выходное слово 1 (бит 03)

```
// Регулирование температуры в четырех точках.
FOR index:=UINT#0 TO UINT#3 BY UINT#1 DO

  // Получение входных значений 1...4.
  CASE index OF
  INT#0:
    AI:=AI1;
  INT#1:
    AI:=AI2;
  INT#2:
    AI:=AI3;
  ELSE
    AI:=AI4;
  END_CASE;

  // Преобразование AI (рег. велич.) в вещественное число.
  PV[index]:=INT_TO_REAL(AI)/REAL#10.0; // CJ1W-PH41U выдает значение, которое в 10
```

```
раз больше фактического, поэтому его нужно поделить на 10,0.

  // Авария по выходу за верхний/нижний предел
  LimitAlarm_REAL_instance[index](
    Enable :=LimitAlarm_ON,
    H :=HighVal[index],
    X :=PV[index],
    L :=LowVal[index],
    EPS :=Hystrs_LimitAlarm[index],
    Q =>Q_LimitAlarm[index],
    QH =>HighAlm[index],
    QL =>LowAlm[index],
    Error =>Error_LimitAlarm[index]);

  // Выдача сигнала для обеспечения безопасности при ошибке в команде LimitAlarmDv_
REAL или при аварии по выходу за верхний/нижний предел.
  Alm_LimitAlarm[index]:=Q_LimitAlarm[index] OR Error_LimitAlarm[index];

  // Авария по верхнему/нижнему отклонению
  LimitAlarmDv_REAL_instance[index](
    Enable :=LimitAlarmDv_ON,
    X :=PV[index],
    H :=DvHighVal[index],
    Y :=SP[index],
    L :=DvLowVal[index],
    EPS :=Hystrs_LimitAlarmDv[index],
    Q =>Q_LimitAlarmDv[index],
    QH =>HighAlmDv[index],
    QL =>LowAlmDv[index],
    Error =>Error_LimitAlarmDv[index]);

  // Выдача сигнала для обеспечения безопасности при ошибке в команде LimitAlarmDv_
REAL или при аварии по выходу за верхний/нижний предел.
  Alm_LimitAlarmDv[index]:=Q_LimitAlarmDv[index] OR Error_LimitAlarmDv[index];

  // Выполнение команд PIDAT.
  PIDAT_instance[index](
    Run :=Run[index],
    ManCtl :=ManCtl[index],
    StartAT :=StartAT[index],
    PV :=PV[index],
    SP :=SP[index],
    OprSetParams :=OprSetParams,
    InitSetParams :=InitSetParams,
    ProportionalBand:=PB[index],
    IntegrationTime :=TI[index],
    DerivativeTime :=TD[index],
    ManMV :=ManMV[index],
    ATDone =>ATDone[index],
    ATBusy =>ATBusy[index],
    Error =>Error_PIDAT[index],
```

```
    ErrorID =>ErrorID[index],
    MV =>MV[index]);

  // Выход широтно-импульсного регулирования
  TimeProportionalOut_instance[index](
    Enable :=TimeProportionalOut_ON,
    AIn :=MV[index],
    CtlPrd :=CtlPrd[index],
    MinPlsWidth :=MinPlsWidth[index],
    Delay :=Delay[index],
    DOut =>DOut_TPO,
    Error =>Error_TimeProportionalOut[index]);

  // Вывод значений в биты 00…03 выходного слова 1.
  CASE index OF
  INT#0:
    DO1:=DOut_TPO;
  INT#1:
    DO2:=DOut_TPO;
  INT#2:
    DO3:=DOut_TPO;
  ELSE
    DO4:=DOut_TPO;
  END_CASE;

END_FOR;
```

LimitAlarm_**

Команда LimitAlarm_** выдает сигнал аварии, если входное значение меньше установленного нижнего предельного значения или больше установленного верхнего предельного значения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LimitAlarm_**	Группа сигнализации аварии выхода за верхний/ нижний предел	FB	LimitAlarm_**_Instance LimitAlarm_** Enable, H, X, L, EPS Q, QH, QL, Error "**" — следует подставить «REAL» или «LREAL».	LimitAlarm_**_instance(Enable, H, X, L, EPS, Q, QH, QL, Error); "**" — следует подставить «REAL» или «LREAL».

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Enable	Активация	Вход	ИСТИНА: выполнить ЛОЖЬ: сброс аварии	Зависит от типа данных.	---	ЛОЖЬ
H	Заданное верхнее предельное значение		Заданный верхний предельный уровень для входного значения			0
X	Входное значение		Контролируемое значение			
L	Заданное нижнее предельное значение		Заданный нижний предельный уровень для входного значения			
EPS	Гистерезис		Гистерезис сигнала аварии	Зависит от типа данных.*1		

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Q	Выход сигнала аварии	Выход	ИСТИНА: произошла авария выхода за верхний или нижний предел. ЛОЖЬ: не произошло аварии выхода за верхний или нижний предел.	Зависит от типа данных.	---	---
QH	Сигнал аварии по выходу за верхний предел		ИСТИНА: произошла авария выхода за верхний предел. ЛОЖЬ: не произошло аварии выхода за верхний предел.			
QL	Сигнал аварии по выходу за нижний предел		ИСТИНА: произошла авария выхода за нижний предел. ЛОЖЬ: не произошло аварии выхода за нижний предел.			

*1. Отрицательные числа исключены.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	ОК																			
H														ОК	ОК					
X	Тип данных должен быть таким же, как у *H*.																			
L	Тип данных должен быть таким же, как у *H*.																			
EPS	Тип данных должен быть таким же, как у *H*.																			
Q	ОК																			
QH	ОК																			
QL	ОК																			

Функция

Команда LimitAlarm_** контролирует, находится ли входное значение в диапазоне между заданными нижним и верхним предельными значениями.
Команда LimitAlarm_** выдает сигнал аварии, если входное значение меньше установленного нижнего предельного значения или больше установленного верхнего предельного значения.
Эту команду можно использовать при регулировании температуры, например, для контроля значения температуры технологического процесса.

Контроль входного значения *X* производится, когда значение *Enable* = ИСТИНА.

Если значение *X* становится больше значения *H* (заданное верхнее предельное значение), выход *QH* (сигнал аварии по выходу за верхний предел) переходит в состояние ИСТИНА.

Если значение *X* становится меньше значения *L* (заданное нижнее предельное значение), выход *QL* (сигнал аварии по выходу за нижний предел) переходит в состояние ИСТИНА.
Если какой-либо из выходов, *QH* или *QL*, находится в состоянии ИСТИНА, выход *Q* также находится в состоянии ИСТИНА.
Пока вход *Enable* находится в состоянии ИСТИНА, значения *X*, *H*, *L* и *EPS* (гистерезис) постоянно обновляются.

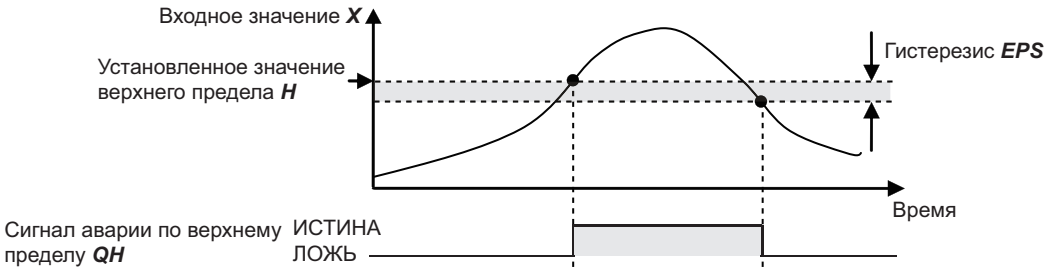
Если вход *Enable* переходит в состояние ЛОЖЬ, сигнал аварии сбрасывается. При сбросе сигнала аварии выходы *Q*, *QH* и *QL* переходят в состояние ЛОЖЬ.

Переменные *H*, *X*, *L* и *EPS* должны иметь тип данных REAL или LREAL.
Имя команды определяется типом данных переменных *H*, *X*, *L* и *EPS*.
Так, если используется команда LimitAlarm_LREAL, то все переменные (*H*, *X*, *L* и *EPS*) имеют тип данных LREAL.

Работа выхода *QH* (сигнал аварии по выходу за верхний предел)

Значение переменной *QH* (сигнал аварии по выходу за верхний предел) изменяется так, как показано ниже.
Чтобы исключить нестабильность состояния сигнала аварии вблизи порога переключения, можно задать гистерезис переключения.

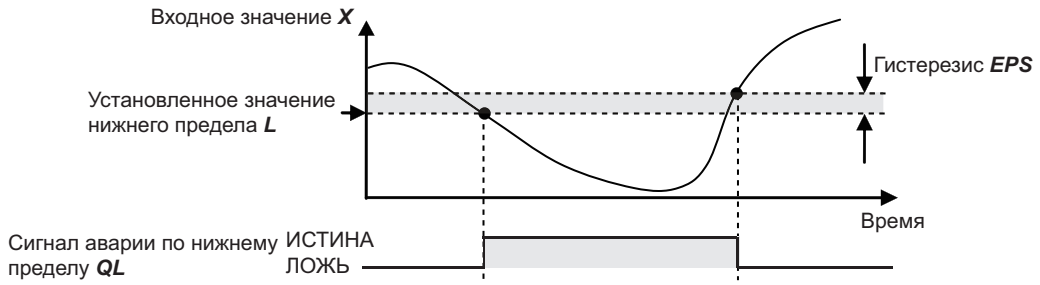
- Если *X* (входное значение) > *H* (заданное верхнее предельное значение), значение = ИСТИНА.
- Если *X* (входное значение) < *H* (заданное верхнее предельное значение) - *EPS* (гистерезис), значение = ЛОЖЬ.



Работа выхода *QL* (сигнал аварии по выходу за нижний предел)

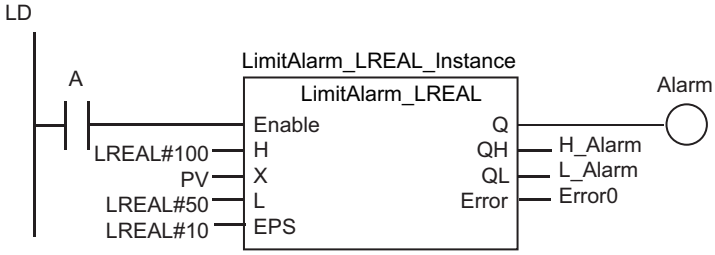
Значение переменной *QL* (сигнал аварии по выходу за нижний предел) изменяется так, как показано ниже.
Чтобы исключить нестабильность состояния сигнала аварии вблизи порога переключения, можно задать гистерезис переключения.

- Если *X* (входное значение) < *L* (заданное нижнее предельное значение), значение = ИСТИНА.
- Если *X* (входное значение) > *L* (заданное нижнее предельное значение) + *EPS* (гистерезис), значение = ЛОЖЬ.



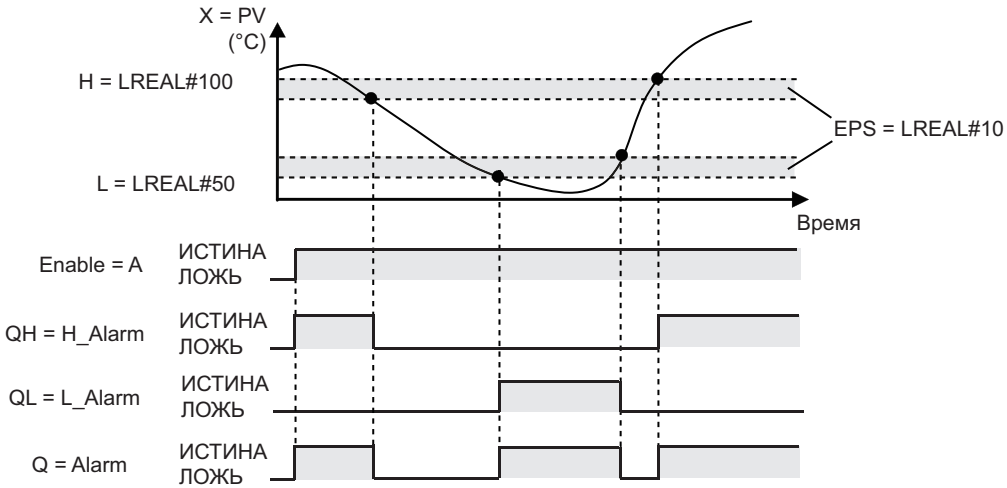
Пример записи

Ниже показан пример записи команды, в котором в переменную *H* (заданное верхнее предельное значение) передается значение 100 °C, в переменную *L* (заданное нижнее предельное значение) передается значение 50 °C, а в переменную *EPS* (гистерезис) передается значение 10 °C.



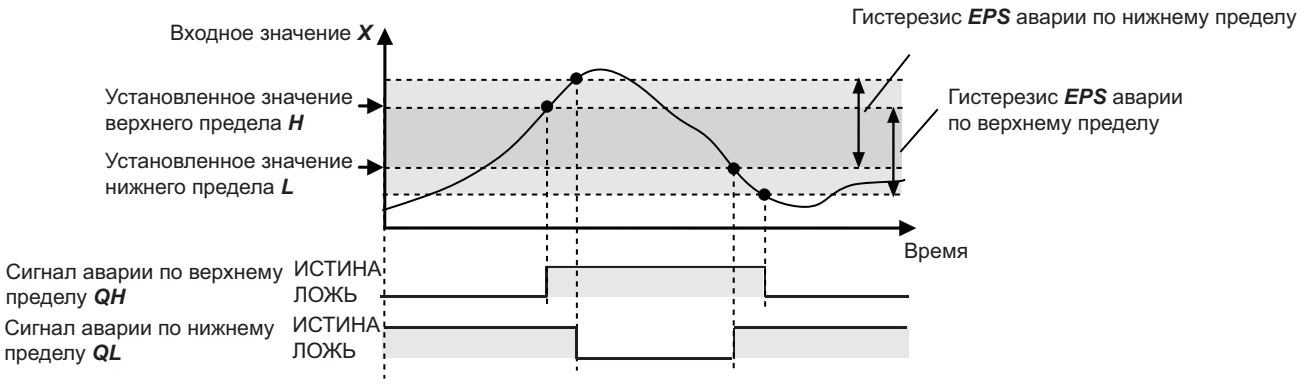
ST

```
LimitAlarm_LREAL_instance(A,LREAL#100,PV,LREAL#50,LREAL#10,Alarm,H_Alarm,L_Alarm,Error0);
```

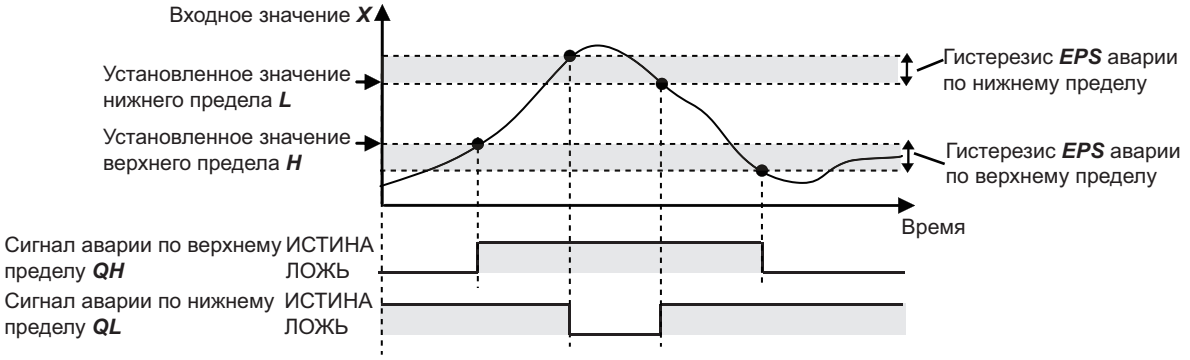


Дополнительная информация

- Чтобы уменьшить время выполнения команды, используйте команду LimitAlarm_REAL.
- Можно настроить параметры следующим образом: *H* - *L* < *EPS*. При такой настройке выходы *QH* и *QL* могут одновременно находиться в состоянии ИСТИНА.



- Можно настроить параметры следующим образом: *H* < *L*. При такой настройке один из выходов, *QH* или *QL*, всегда будет находиться в состоянии ИСТИНА.



Меры предосторожности для обеспечения надлежащей эксплуатации

- Выход значения *EPS* за пределы допустимого диапазона приводит к возникновению ошибки. Выход *Error* переходит в состояние ИСТИНА, а выходы *Q*, *QH* и *QL* переходят в состояние ЛОЖЬ.
- Эту команду можно использовать в качестве меры обеспечения безопасности, например, для отключения выхода регулирования температуры при выдаче сигнала аварии. В этом случае, однако, следует продумать весь комплекс мер безопасности, чтобы переключение выходов *Q*, *QH* и *QL* в состояние ЛОЖЬ из-за возникновения ошибки не создавало опасных ситуаций в системе. Практический пример см. в разделе *Пример программы* на стр. 2-845 для команды TimeProportionalOut.

Пример программы

См. *Пример программы* на стр. 2-845 для команды TimeProportionalOut.

LimitAlarmDv_**

Команда LimitAlarmDv_** выдает сигнал аварии, если отклонение входного значения от опорного значения превышает заданное значение нижнего отклонения или заданное значение верхнего отклонения.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LimitAlarmDv_**	Группа сигнализации аварии верхнего/ нижнего отклонения	FB	LimitAlarmDv_**_Instance LimitAlarmDv_** Enable X H Y L EPS Q QH QL Error "**" — следует подставить «REAL» или «LREAL».	LimitAlarmDv_**instance(Enable, X, H, Y, L, EPS, Q, QH, QL, Error); "**" — следует подставить «REAL» или «LREAL».

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Enable	Активация	Вход	ИСТИНА: выполнить ЛОЖЬ: сброс аварии	Зависит от типа данных.	---	ЛОЖЬ
X	Входное значение		Контролируемое значение			0
H	Заданное значение верхнего отклонения		Предельное отклонение входного контролируемого значения от опорного значения в сторону увеличения, при котором подается сигнал аварии.			
Y	Опорное значение		Опорное значение, отклонение относительно которого контролируется.			
L	Заданное значение нижнего отклонения		Предельное отклонение входного контролируемого значения от опорного значения в сторону уменьшения, при котором подается сигнал аварии.			
EPS	Гистерезис		Гистерезис сигнала аварии	Зависит от типа данных.*1		
Q	Выход сигнала аварии отклонения	Выход	ИСТИНА: произошла авария верхнего или нижнего отклонения. ЛОЖЬ: не произошло аварии ни верхнего, ни нижнего отклонения.	Зависит от типа данных.	---	---
QH	Сигнал аварии по верхнему отклонению		ИСТИНА: произошла авария верхнего отклонения. ЛОЖЬ: не произошло аварии верхнего отклонения.			
QL	Сигнал аварии по нижнему отклонению		ИСТИНА: произошла авария нижнего отклонения. ЛОЖЬ: не произошло аварии нижнего отклонения.			

*1. Отрицательные числа исключены.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	ОК																			
X														ОК	ОК					
H	Тип данных должен быть таким же, как у *X*.																			
Y	Тип данных должен быть таким же, как у *X*.																			
L	Тип данных должен быть таким же, как у *X*.																			
EPS	Тип данных должен быть таким же, как у *X*.																			
Q	ОК																			
QH	ОК																			
QL	ОК																			

Функция

Команда LimitAlarmDv_** контролирует, не отклоняется ли входное значение от опорного значения больше, чем на заданное значение нижнего отклонения или заданное значение верхнего отклонения.
Если величина отклонения превосходит заданное значение нижнего отклонения или заданное значение верхнего отклонения, команда выдает соответствующий сигнал аварии.
Эту команду можно использовать при регулировании температуры, например, для контроля отклонения температуры технологического процесса от заданного значения.

Контроль отклонения входного значения *X* от опорного значения *Y* производится, когда значение *Enable* = ИСТИНА.
Если отклонение *X* от *Y* в сторону увеличения («верхнее отклонение») становится больше значения *H* (заданное значение верхнего отклонения), выход *QH* (сигнал аварии по верхнему отклонению) переходит в состояние ИСТИНА.
Если отклонение *X* от *Y* в сторону уменьшения («нижнее отклонение») становится больше значения *L* (заданное значение нижнего отклонения), выход *QL* (сигнал аварии по нижнему отклонению) переходит в состояние ИСТИНА.
Если какой-либо из выходов, *QH* или *QL*, находится в состоянии ИСТИНА, выход *Q* также находится в состоянии ИСТИНА.
Пока вход *Enable* находится в состоянии ИСТИНА, значения *X*, *H*, *Y*, *L* и *EPS* (гистерезис) постоянно обновляются.

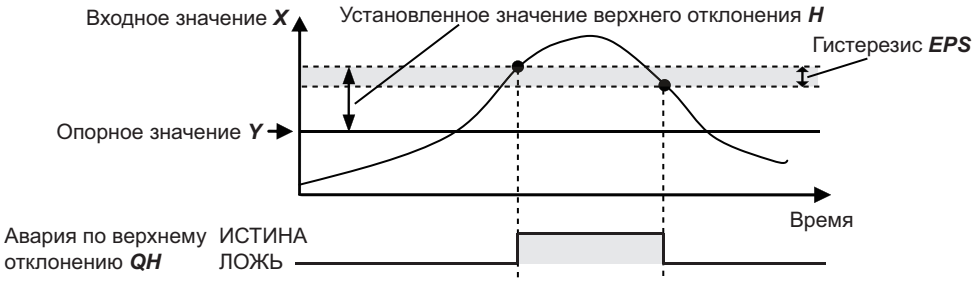
Если вход *Enable* переходит в состояние ЛОЖЬ, сигнал аварии сбрасывается. При сбросе сигнала аварии выходы *Q*, *QH* и *QL* переходят в состояние ЛОЖЬ.

Переменные *X*, *H*, *Y*, *L* и *EPS* должны иметь тип данных REAL или LREAL.
Имя команды определяется типом данных переменных *X*, *H*, *Y*, *L* и *EPS*.
Так, если используется команда LimitAlarmDv_LREAL, то все переменные (*X*, *H*, *Y*, *L* и *EPS*) имеют тип данных LREAL.

Работа выхода *QH* (сигнал аварии по верхнему отклонению)

Выход *QH* (сигнал аварии по верхнему отклонению) служит для подачи сигнала аварии в случае, когда входное значение чрезмерно отклоняется от заданного опорного значения *Y* в сторону увеличения. Значение переменной *QH* изменяется так, как показано ниже. Чтобы исключить нестабильность состояния сигнала аварии вблизи порога переключения, можно задать гистерезис переключения.

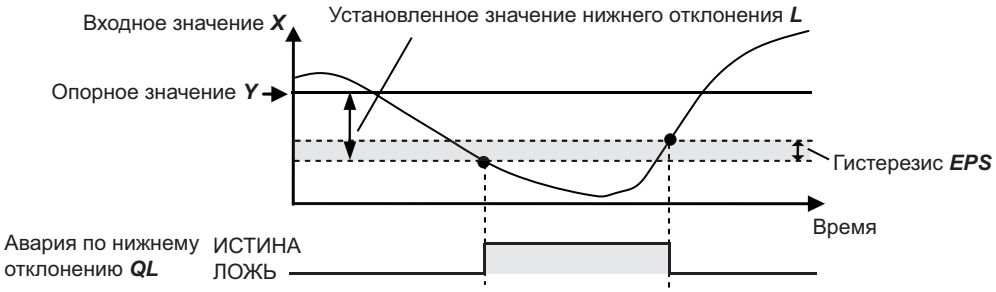
- Если *X* (входное значение) - *Y* (опорное значение) > *H* (заданное значение верхнего отклонения), то значение = ИСТИНА.
- Если *X* (входное значение) - *Y* (опорное значение) < *H* (заданное значение верхнего отклонения) - *EPS* (гистерезис), то значение = ЛОЖЬ.



Работа выхода *QL* (сигнал аварии по нижнему отклонению)

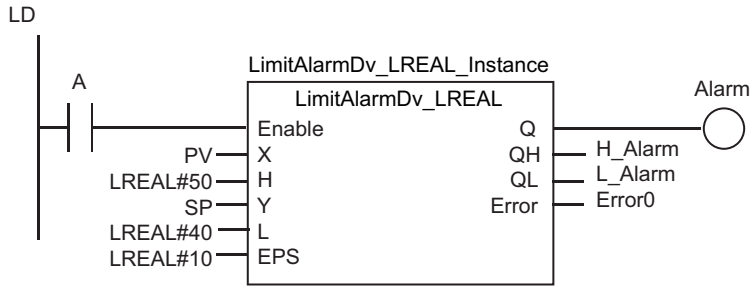
Выход *QL* (сигнал аварии по нижнему отклонению) служит для подачи сигнала аварии в случае, когда входное значение чрезмерно отклоняется от заданного опорного значения *Y* в сторону уменьшения. Значение переменной *QL* изменяется так, как показано ниже. Чтобы исключить нестабильность состояния сигнала аварии вблизи порога переключения, можно задать гистерезис переключения.

- Если -(*X* (входное значение) - *Y* (опорное значение)) > *L* (заданное значение нижнего отклонения), то значение = ИСТИНА.
- Если -(*X* (входное значение) - *Y* (опорное значение)) < *L* (заданное значение нижнего отклонения) - *EPS* (гистерезис), то значение = ЛОЖЬ.



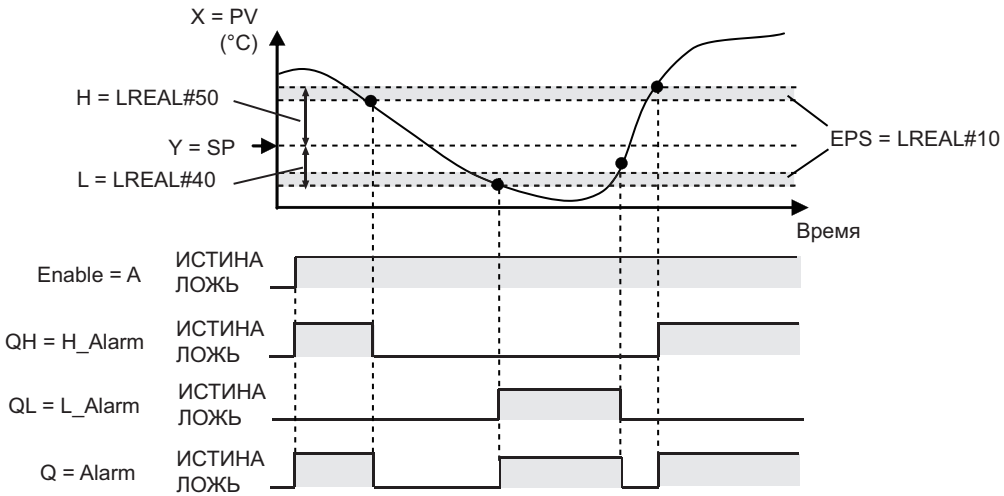
Пример записи

Ниже показан пример записи команды, в котором в переменную *H* (заданное значение верхнего отклонения) передается значение 50 °C, в переменную *L* (заданное значение нижнего отклонения) передается значение 40 °C, а в переменную *EPS* (гистерезис) передается значение 10 °C.



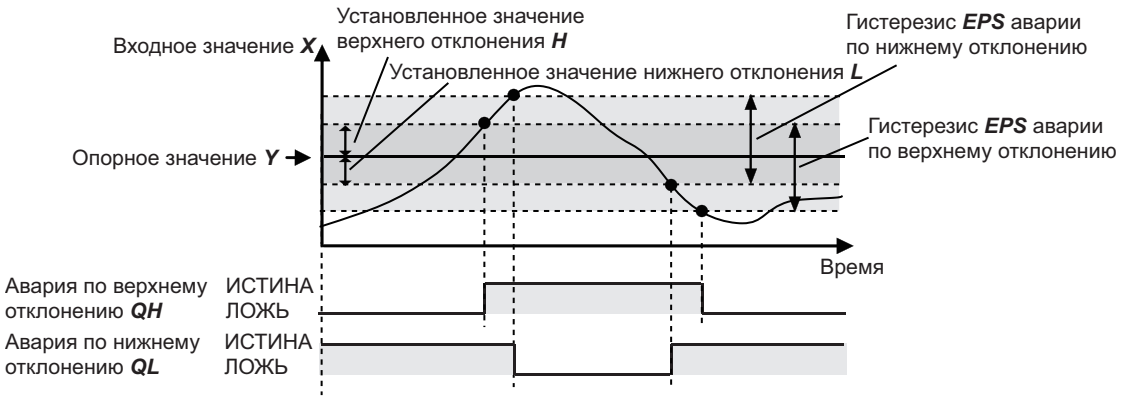
ST

```
LimitAlarmDv_LREAL_instance(A,PV,LREAL#50,SP,LREAL#40,LREAL#10,Alarm,H_Alarm,L_Alarm,Error0);
```

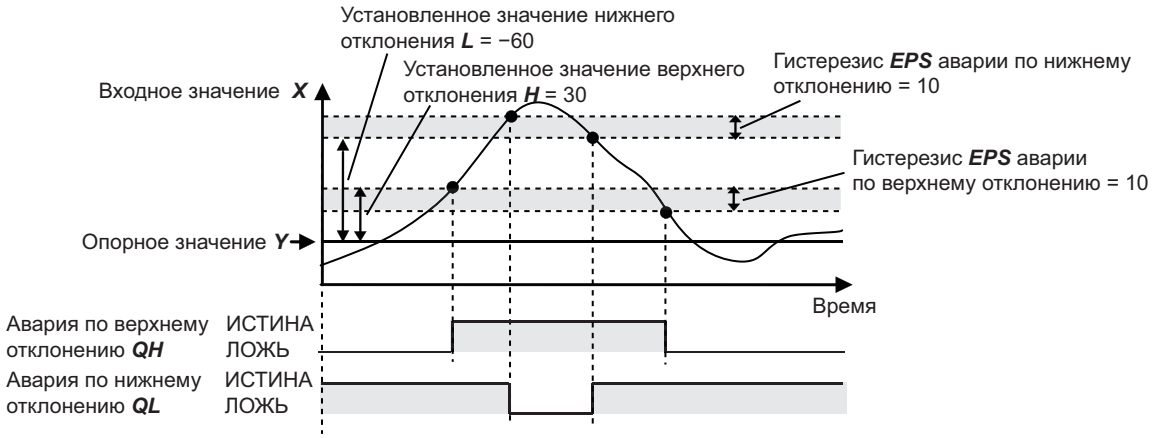


Дополнительная информация

- Чтобы уменьшить время выполнения команды, используйте команду LimitAlarmDv_REAL.
- В переменную *EPS* можно ввести значение, которое меньше суммы *H* + *L*. При такой настройке выходы *QH* и *QL* могут одновременно находиться в состоянии ИСТИНА.



- Можно задать значения, при которых сумма *H* + *L* < 0. При такой настройке один из выходов, *QH* или *QL*, всегда будет находиться в состоянии ИСТИНА. На рисунке ниже показан пример работы команды для случая, когда значение *L* = -60, а значение *H* = 30.



Меры предосторожности для обеспечения надлежащей эксплуатации

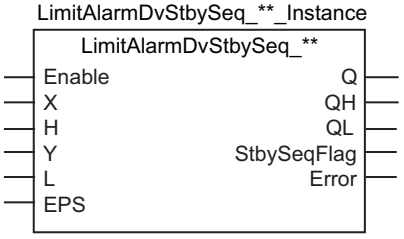
- Выход значения *EPS* за пределы допустимого диапазона приводит к возникновению ошибки. Выход *Error* переходит в состояние ИСТИНА, а выходы *Q*, *QH* и *QL* переходят в состояние ЛОЖЬ.
- Эту команду можно использовать в качестве меры обеспечения безопасности, например, для отключения выхода регулирования температуры при выдаче сигнала аварии по отклонению. В этом случае, однако, следует продумать весь комплекс мер безопасности, чтобы переключение выходов *Q*, *QH* и *QL* в состояние ЛОЖЬ из-за возникновения ошибки не создавало опасных ситуаций в системе. Практический пример см. в разделе *Пример программы* на стр. 2-845 для команды TimeProportionalOut.

Пример программы

См. *Пример программы* на стр. 2-845 для команды TimeProportionalOut.

LimitAlarmDvStbySeq_**

Команда LimitAlarmDvStbySeq_** выдает сигналы аварии по нижнему и верхнему отклонению с контролем последовательности событий.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
LimitAlarmDv StbySeq_**	Группа сигнализации аварии верхнего/ нижнего отклонения с контролем последовательности	FB	 "**" — следует подставить «REAL» или «LREAL».	LimitAlarmDvStbySeq_**_instance(Enable, X, H, Y, L, EPS, Q, QH, QL, StbySeqFlag, Error); "**" — следует подставить «REAL» или «LREAL».

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Enable	Активация	Вход	ИСТИНА: выполнить ЛОЖЬ: сброс аварии	Зависит от типа данных.	---	ЛОЖЬ
X	Входное значение		Значение, для которого формируется сигнал аварии отклонения			
H	Заданное значение верхнего отклонения		Предельное отклонение входного контролируемого значения от опорного значения в сторону увеличения, при котором подается сигнал аварии.			0
Y	Опорное значение		Опорное значение, отклонение относительно которого контролируется.			
L	Заданное значение нижнего отклонения		Предельное отклонение входного контролируемого значения от опорного значения в сторону уменьшения, при котором подается сигнал аварии.			
EPS	Гистерезис		Гистерезис сигнала аварии	Зависит от типа данных.*1		
Q	Выход сигнала аварии отклонения	Выход	ИСТИНА: произошла авария верхнего или нижнего отклонения. ЛОЖЬ: не произошло аварии ни верхнего, ни нижнего отклонения.	Зависит от типа данных.	---	---
QH	Сигнал аварии по верхнему отклонению		ИСТИНА: произошла авария верхнего отклонения. ЛОЖЬ: не произошло аварии верхнего отклонения.			
QL	Сигнал аварии по нижнему отклонению		ИСТИНА: произошла авария нижнего отклонения. ЛОЖЬ: не произошло аварии нижнего отклонения.			
StbySeqFlag	Флаг контроля последовательности событий		ИСТИНА: активировано ЛОЖЬ: деактивировано			

*1. Отрицательные числа исключены.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	ОК																			
X														ОК	ОК					
H	Тип данных должен быть таким же, как у *X*.																			
Y	Тип данных должен быть таким же, как у *X*.																			
L	Тип данных должен быть таким же, как у *X*.																			
EPS	Тип данных должен быть таким же, как у *X*.																			
Q	ОК																			
QH	ОК																			
QL	ОК																			
StbySeqFlag	ОК																			

Функция

Команда LimitAlarmDvStbySeq_** контролирует, не отклоняется ли входное значение от опорного значения больше, чем на заданное значение нижнего отклонения или заданное значение верхнего отклонения.
Если величина отклонения превосходит заданное значение нижнего отклонения или заданное значение верхнего отклонения, команда выдает соответствующий сигнал аварии. Однако сигнал аварии не выдается до тех пор, пока первоначальное отклонение входного значения от опорного значения не оказывается в пределах допустимого (т. е. пока входное значение не оказывается внутри области между заданными предельными значениями верхнего и нижнего отклонения).
Эту команду можно использовать при регулировании температуры, например, для контроля отклонения температуры технологического процесса от заданного значения, когда требуется исключить подачу сигнала аварии до стабилизации температуры.

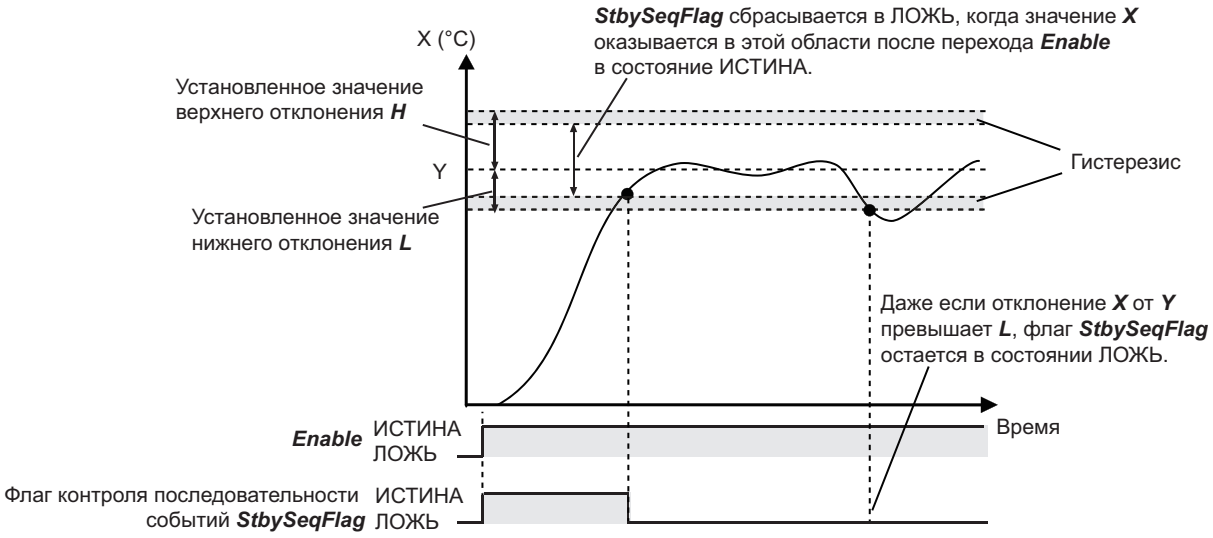
Контроль отклонения входного значения *X* от опорного значения *Y* производится, когда значение *Enable* = ИСТИНА. Однако отклонение не контролируется, пока флаг *StbySeqFlag* (флаг контроля последовательности событий) находится в состоянии ИСТИНА.
Если отклонение *X* от *Y* в сторону увеличения («верхнее отклонение») становится больше значения *H* (заданное значение верхнего отклонения), выход *QH* (сигнал аварии по верхнему отклонению) переходит в состояние ИСТИНА.
Если отклонение *X* от *Y* в сторону уменьшения («нижнее отклонение») становится больше значения *L* (заданное значение нижнего отклонения), выход *QL* (сигнал аварии по нижнему отклонению) переходит в состояние ИСТИНА.
Если какой-либо из выходов, *QH* или *QL*, находится в состоянии ИСТИНА, выход *Q* также находится в состоянии ИСТИНА.
Пока вход *Enable* находится в состоянии ИСТИНА, значения *X*, *H*, *Y*, *L* и *EPS* постоянно обновляются.

Если вход *Enable* переходит в состояние ЛОЖЬ, сигнал аварии сбрасывается. При сбросе сигнала аварии выходы *Q*, *QH* и *QL*, а также флаг *StbySeqFlag* сбрасываются в состояние ЛОЖЬ.

Флаг *StbySeqFlag* сбрасывается в состояние ЛОЖЬ, если после перехода входа *Enable* в состояние ИСТИНА оказываются соблюдены все указанные ниже условия.

После перехода в состояние ЛОЖЬ флаг *StbySeqFlag* не перейдет в состояние ИСТИНА до тех пор, пока вход *Enable* вновь не переключится из состояния ЛОЖЬ в состояние ИСТИНА.

- *X* (входное значение) - *Y* (опорное значение) < *H* (заданное значение верхнего отклонения) - *EPS* (гистерезис)
- -(*X* (входное значение) - *Y* (опорное значение)) < *L* (заданное значение нижнего отклонения) - *EPS* (гистерезис)

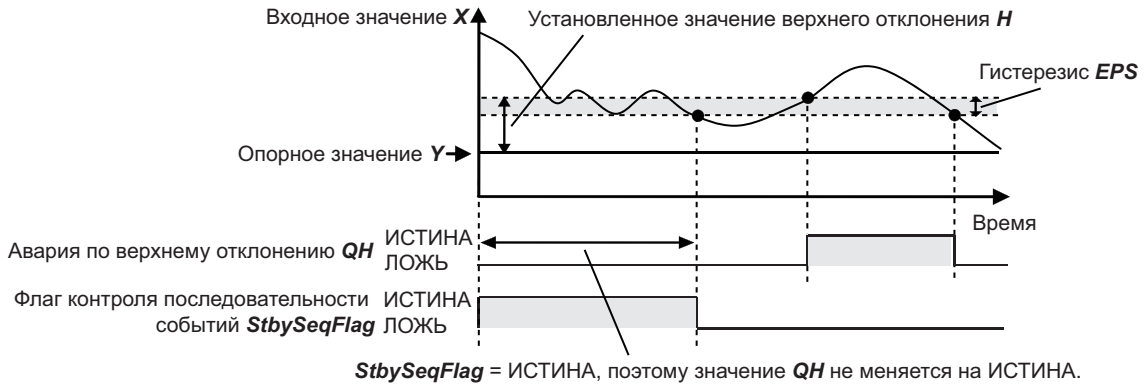


Переменные *X*, *H*, *Y*, *L* и *EPS* должны иметь тип данных REAL или LREAL.
Имя команды определяется типом данных переменных *X*, *H*, *Y*, *L* и *EPS*.
Так, если используется команда LimitAlarmDvStbySeq_LREAL, то все переменные (*X*, *H*, *Y*, *L* и *EPS*) имеют тип данных LREAL.

Работа выхода *QH* (сигнал аварии по верхнему отклонению)

Выход *QH* (сигнал аварии по верхнему отклонению) служит для подачи сигнала аварии в случае, когда входное значение чрезмерно отклоняется от заданного опорного значения *Y* в сторону увеличения.
Пока флаг *StbySeqFlag* = ЛОЖЬ, значение *QH* изменяется так, как показано ниже.
Чтобы исключить нестабильность состояния сигнала аварии вблизи порога переключения, можно задать гистерезис переключения.

- Если *X* (входное значение) - *Y* (опорное значение) > *H* (заданное значение верхнего отклонения), то значение = ИСТИНА.
- Если *X* (входное значение) - *Y* (опорное значение) < *H* (заданное значение верхнего отклонения) - *EPS* (гистерезис), то значение = ЛОЖЬ.



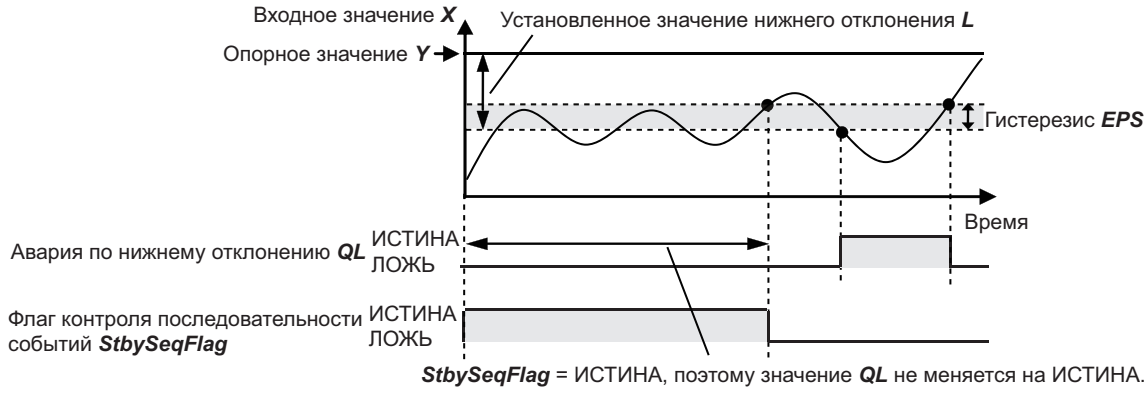
Работа выхода *QL* (сигнал аварии по нижнему отклонению)

Выход *QL* (сигнал аварии по нижнему отклонению) служит для подачи сигнала аварии в случае, когда входное значение чрезмерно отклоняется от заданного опорного значения *Y* в сторону уменьшения.

Пока *StbySeqFlag* = ЛОЖЬ, значение *QL* изменяется так, как показано ниже.

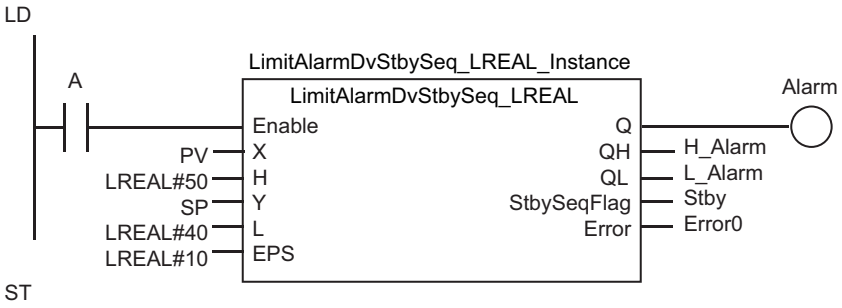
Чтобы исключить нестабильность состояния сигнала аварии вблизи порога переключения, можно задать гистерезис переключения.

- Если -(*X* (входное значение) - *Y* (опорное значение)) > *L* (заданное значение нижнего отклонения), то значение = ИСТИНА.
- Если -(*X* (входное значение) - *Y* (опорное значение)) < *L* (заданное значение нижнего отклонения) - *EPS* (гистерезис), то значение = ЛОЖЬ.



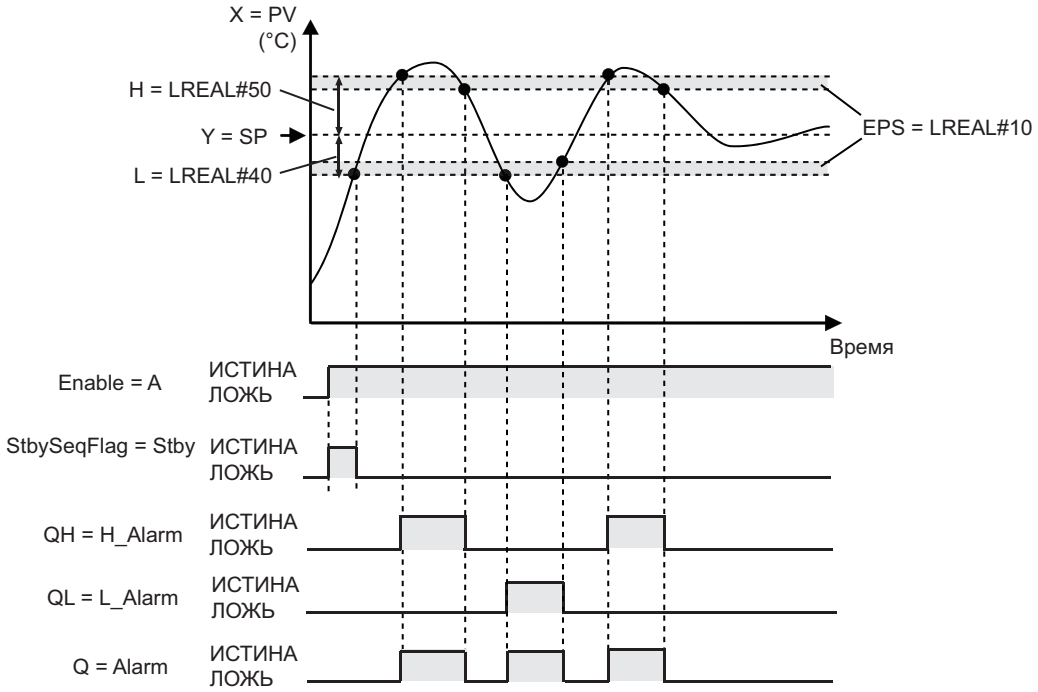
Пример записи

Ниже показан пример записи команды, в котором в переменную *H* (заданное значение верхнего отклонения) передается значение 50 °C, в переменную *L* (заданное значение нижнего отклонения) передается значение 40 °C, а в переменную *EPS* (гистерезис) передается значение 10 °C.



ST

```
LimitAlarmDvStbySeq_LREAL_Instance(A,PV,LREAL#50,SP,LREAL#40,LREAL#10,Alarm,H_Alarm,L_Alarm,Stby,Error0);
```



Дополнительная информация

- Чтобы уменьшить время выполнения команды, используйте команду LimitAlarmDvStbySeq_REAL.
- В переменную *EPS* можно ввести значение, которое больше суммы *H* + *L*. При такой настройке выходы *QH* и *QL* могут одновременно находиться в состоянии ИСТИНА. См. описание команды *LimitAlarmDv_*** на стр. 2-865.
- Можно настроить параметры следующим образом: *H* + *L* < 0. При такой настройке один из выходов, *QH* или *QL*, всегда будет находиться в состоянии ИСТИНА, пока флаг *StbySeqFlag* = ЛОЖЬ. См. описание команды *LimitAlarmDv_*** на стр. 2-865.

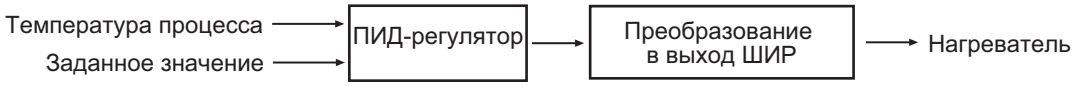
Меры предосторожности для обеспечения надлежащей эксплуатации

- Выход значения *EPS* за пределы допустимого диапазона приводит к возникновению ошибки. Выход *Error* переходит в состояние ИСТИНА, а выходы *Q*, *QH* и *QL* переходят в состояние ЛОЖЬ.
- Эту команду можно использовать в качестве меры обеспечения безопасности, например, для отключения выхода регулирования температуры при выдаче сигнала аварии по отклонению. В этом случае, однако, следует продумать весь комплекс мер безопасности, чтобы

переключение выходов *Q*, *QH* и *QL* в состояние ЛОЖЬ из-за возникновения ошибки не создавало опасных ситуаций в системе. Пример применения команды см. в разделе *Пример программы* на стр. 2-877.

Пример программы

Рассмотрим пример программы, которая осуществляет регулирование температуры в четырех точках, подает аварийные сигналы при выходе температуры за верхнюю или нижнюю предельные границы, а также сигнализирует чрезмерное отклонение температуры в сторону увеличения или в сторону уменьшения, контролируя при этом последовательность событий.
Применяется ПИД-регулирование. Получаемые в результате ПИД-регулирования управляющие воздействия преобразуются в импульсные сигналы для широтно-импульсного регулирования, которые подаются на нагревательные устройства.



Характеристики

В таблице ниже указано используемое оборудование и параметры регулирования температуры.

Устройство/параметр	Характеристики
Модуль входов	CJ1W-PH41U (модуль универсальных входов с гальванической развязкой)
Типы входов	Термопары типа K
Модуль выходов	CJ1W-OD212 (модуль транзисторных выходов)
Заданное значение	100 °C
Верхняя предельная температура	200 °C
Нижняя предельная температура	0 °C
Гистерезис аварии по выходу за верхний/нижний предел	5 °C
Верхнее отклонение температуры	50 °C
Нижнее отклонение температуры	50 °C
Гистерезис аварии по верхнему/нижнему отклонению	3 °C
Период измерения для ПИД-регулирования	100 мс
Период выходного сигнала управления	1 с

Конфигурация и настройки

В таблице ниже приведены используемые параметры модуля входов CJ1W-PH41U.

Параметр	Заданное значение
Input1: тип входного сигнала	K(1)
Input2: тип входного сигнала	K(1)
Input3: тип входного сигнала	K(1)
Input4: тип входного сигнала	K(1)

В таблице ниже перечислены используемые параметры карты соответствия входов и выходов (I/O map).

Модуль	Порт ввода-вывода	Описание	Переменная
CJ1W-PH41U	Ch1_AIInPV	Вход 1, регулируемая величина (значение типа INT)	AI1
	Ch2_AIInPV	Вход 2, регулируемая величина (значение типа INT)	AI2
	Ch3_AIInPV	Вход 3, регулируемая величина (значение типа INT)	AI3
	Ch4_AIInPV	Вход 4, регулируемая величина (значение типа INT)	AI4
CJ1W-OD212	Ch1_Out00	Бит 00 выходного слова 1	DO1
	Ch1_Out01	Бит 01 выходного слова 1	DO2
	Ch1_Out02	Бит 02 выходного слова 1	DO3
	Ch1_Out03	Бит 03 выходного слова 1	DO4

Ниже перечислены входы и выходы, которые используются для регулирования температуры в четырех точках.

Вход	Выход
AI1	DO1
AI2	DO2
AI3	DO3
AI4	DO4

Период выполнения задачи, которой назначена программа, равен 1 мс.

● Структурная схема

См. *Пример программы* на стр. 2-845 для команды TimeProportionalOut.

Порядок выполнения операций

Для каждой из четырех точек измерения соблюдается следующий порядок операций.

1 Считывается температура процесса.

2 Используется команда LimitAlarm_REAL для выдачи сигналов аварии по выходу за верхний/нижний предел для температуры процесса.

3 В случае, если возникла ошибка в команде LimitAlarm_REAL или подан сигнал аварии по выходу за верхний или нижний предел, выдается сигнал для обеспечения безопасности.

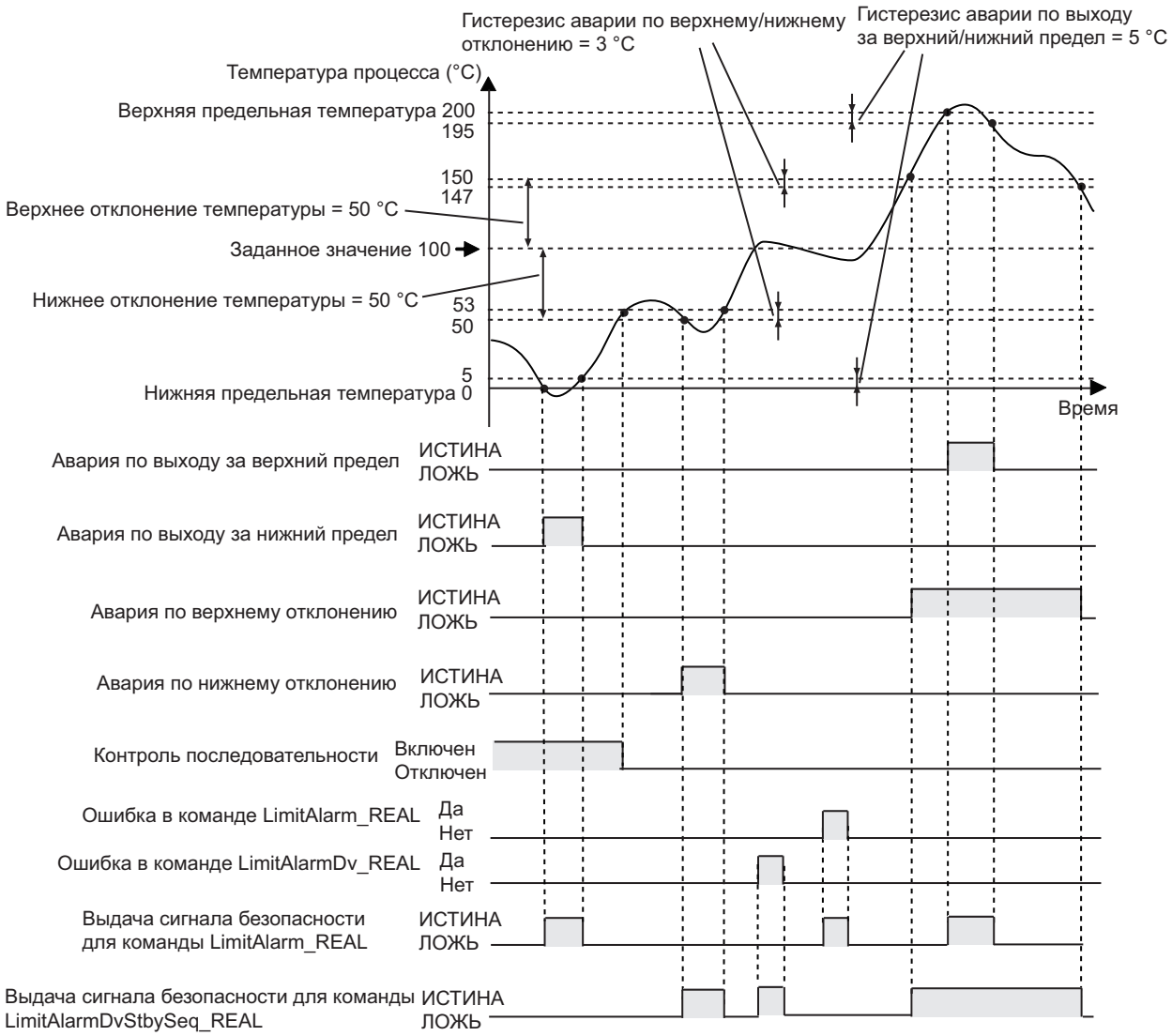
4 Используется команда LimitAlarmDvStbySeq_REAL для выдачи сигналов аварии по верхнему/нижнему отклонению температуры процесса от заданного значения с контролем последовательности событий.

5 В случае, если возникла ошибка в команде LimitAlarmDvStbySeq_REAL или подан сигнал аварии по верхнему/нижнему отклонению, выдается сигнал для обеспечения безопасности.

6 Выполняется регулирование температуры с использованием команды PIDAT.

7 Используется команда TimeProportionalOut для преобразования управляющего воздействия с выхода ПИД-регулятора в сигнал широтно-импульсного регулирования, подаваемый на нагревательное устройство.

● Формирование сигналов аварии по выходу за верхний/нижний предел и аварии по верхнему/нижнему отклонению с контролем последовательности событий



Определения глобальных переменных

● Глобальные переменные

Переменная	Тип данных	Параметр «AT»*1	Комментарий
AI1	INT	IOBus://rack#0/slot#0/Ch1_AIInPV	Вход 1, регулируемая величина (значение типа INT)
AI2	INT	IOBus://rack#0/slot#0/Ch2_AIInPV	Вход 2, регулируемая величина (значение типа INT)
AI3	INT	IOBus://rack#0/slot#0/Ch3_AIInPV	Вход 3, регулируемая величина (значение типа INT)
AI4	INT	IOBus://rack#0/slot#0/Ch4_AIInPV	Вход 4, регулируемая величина (значение типа INT)
DO1	BOOL	IOBus://rack#0/slot#1/Ch1_Out/Ch1_Out00	Бит 00 выходного слова 1

Переменная	Тип данных	Параметр «AT»*1	Комментарий
DO2	BOOL	IOBus://rack#0/slot#1/Ch1_Out/Ch1_Out01	Бит 01 выходного слова 1
DO3	BOOL	IOBus://rack#0/slot#1/Ch1_Out/Ch1_Out02	Бит 02 выходного слова 1
DO4	BOOL	IOBus://rack#0/slot#1/Ch1_Out/Ch1_Out03	Бит 03 выходного слова 1

*1. В таблице приведены переменные для модуля входов CJ1W-PH41U, установленного в слот №0 стойки №0, и для модуля выходов CJ1W-OD212, установленного в слот №1 той же стойки.

Примечание Глобальные переменные для порта каждого модуля генерируются автоматически на основе настроенного соответствия входов-выходов.

Программа на языке LD

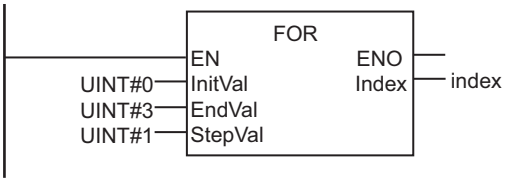
Имя	Тип данных	По умолчанию	Сохранение	Комментарий
index	UINT	0	☐	Счетчик цикла
LimitAlarm_ON	BOOL	Истина	☐	Выполнение команды аварии выхода за верхний/нижний предел
LimitAlarmDvStbySeq_ON	BOOL	Истина	☐	Выполнение команды аварии верхнего/нижнего отклонения с контролем последовательности
TimeProportionalOut_ON	BOOL	Истина	☐	Выполнение команды TimeProportionalOut
AI	INT	0	☐	Текущее значение
PV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Регулируемая величина
SP	ARRAY[0..3] OF REAL	[4(100)]	☐	Заданное значение
DOut_TPO	BOOL	Ложь	☐	Выход широтно-импульсного регулирования
HighVal	ARRAY[0..3] OF REAL	[4(200)]	☐	Установленный верхний предел для сигнализации аварии по выходу за верхний/нижний предел

Имя	Тип данных	По умолчанию	Сохранение	Комментарий
LowVal	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Установленный нижний предел для сигнализации аварии по выходу за верхний/нижний предел
Hystrs_LimitAlarm	ARRAY[0..3] OF REAL	[4(5)]	☐	Гистерезис аварии по выходу за верхний/нижний предел
Q_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выход сигнала аварии по выходу за верхний/нижний предел
HighAlm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по выходу за верхний предел
LowAlm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по выходу за нижний предел
Error_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде LimitAlarm_REAL
Alm_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал для обеспечения безопасности для команды аварии выхода за верхний/нижний предел
DvHighVal	ARRAY[0..3] OF REAL	[4(50)]	☐	Установленное верхнее отклонение для сигнализации аварии по верхнему/нижнему отклонению
DvLowVal	ARRAY[0..3] OF REAL	[4(50)]	☐	Установленное нижнее отклонение для сигнализации аварии по верхнему/нижнему отклонению
Q_LimitAlarmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выход сигнала аварии по верхнему/нижнему отклонению
HighAlmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по верхнему отклонению
LowAlmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по нижнему отклонению
StbySeqFlag	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Флаг контроля последовательности событий

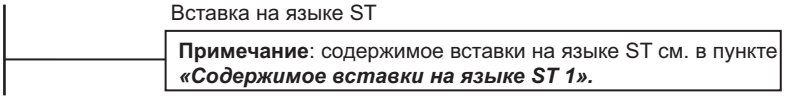
Имя	Тип данных	По умолчанию	Сохранение	Комментарий
Error_LimitAlarmDvStbySeq	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде LimitAlarmDvStbySeq_REAL
Hystrs_LimitAlarmDv	ARRAY[0..3] OF REAL	[4(3)]	☐	Гистерезис аварии по верхнему/нижнему отклонению
Alm_LimitAlarmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал для обеспечения безопасности для команды аварии по верхнему/нижнему отклонению
Run	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Условие выполнения
ManCtl	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ручное/автоматическое управление
StartAT	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Условие выполнения автонастройки
OprSetParams	_sOPR_SET_PARAMS	(MVLowLmt:=0.0, MVUpLmt:=100.0, ManResetVal:=0.0, MVTrackSw:=False, MVTrackVal:=0.0, StopMV:=0.0, ErrorMV:=0.0,Alpha:=0.65, ATCalcGain:=1.0, ATHystrs:=0.2)	☐	Рабочие параметры
InitSetParams	_sINIT_SET_PARAMS	(SampTime:=T#100 ms, RngLowLmt:=-10.0,RngUpLmt:=1000.0, DirOpr:=False)	☐	Начальные параметры
PB	ARRAY[0..3] OF REAL	[4(10)]	☑	Зона пропорциональности
TI	ARRAY[0..3] OF TIME	[4(T#0 c)]	☑	Постоянная времени интегрирования
TD	ARRAY[0..3] OF TIME	[4(T#0 c)]	☑	Постоянная времени дифференцирования
ManMV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Ручное управляющее воздействие
ATDone	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Нормальное завершение автонастройки
ATBusy	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выполнение автонастройки
Error_PIDAT	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде PIDAT
ErrorID	ARRAY[0..3] OF WORD	[4(16#0)]	☐	Идентификатор ошибки для команды PIDAT
MV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Управляющее воздействие
CtlPrd	ARRAY[0..3] OF TIME	[4(T#1 c)]	☐	Период регулирования
MinPlsWidth	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Минимальная ширина импульса

Имя	Тип данных	По умолчанию	Сохранение	Комментарий
Delay	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Время задержки включения
Error_TimeProportionalOut	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде TimeProportionalOut
LimitAlarm_REAL_instance	LimitAlarm_REAL		☐	
LimitAlarmDvStbySeq_REAL_instance	LimitAlarmDvStbySeq_REAL		☐	
PIDAT_instance	PIDAT		☐	
TimeProportionalOut_instance	TimeProportionalOut		☐	

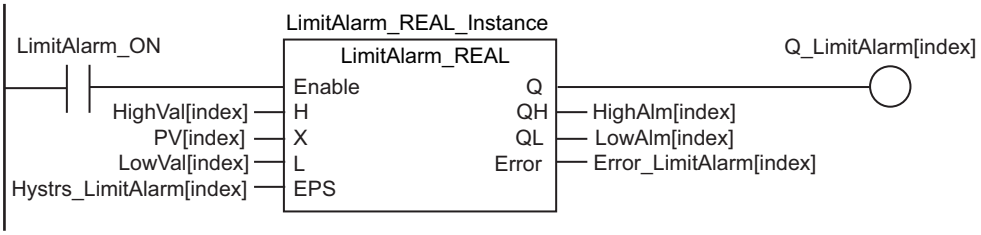
Регулирование температуры в четырех точках.



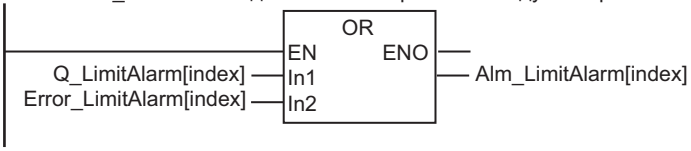
Получение значения регулируемой величины.



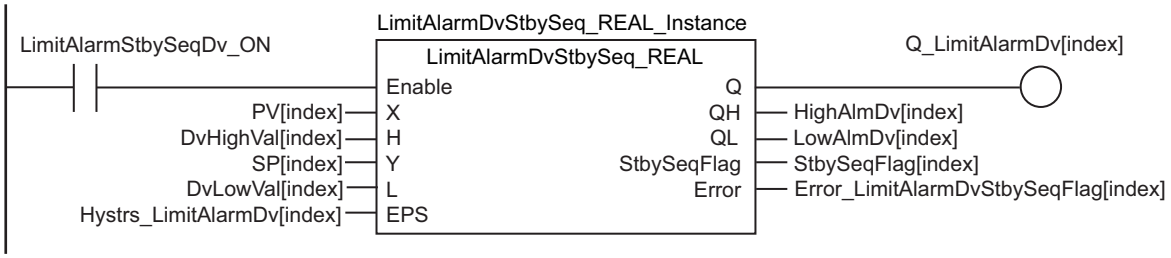
Сигнализация аварии по выходу за верхний или нижний предел



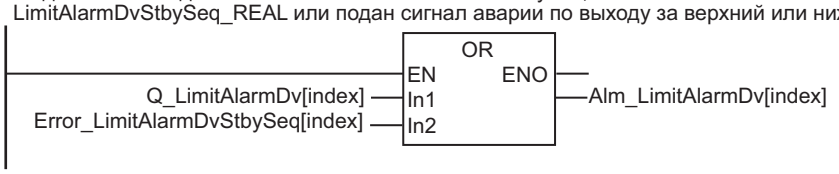
Выдача сигнала для обеспечения безопасности в случае, если возникла ошибка в команде LimitAlarm_REAL или подан сигнал аварии по выходу за верхний или нижний предел.



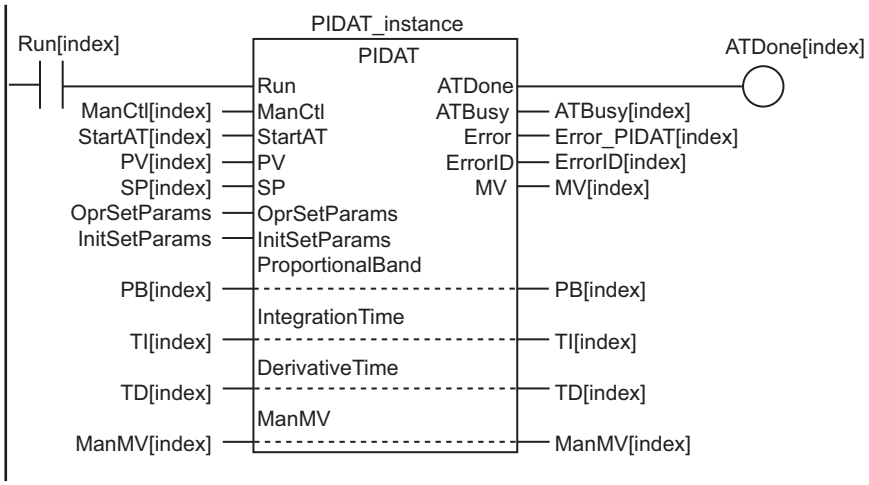
Сигнализация аварии верхнего/нижнего отклонения с контролем последовательности



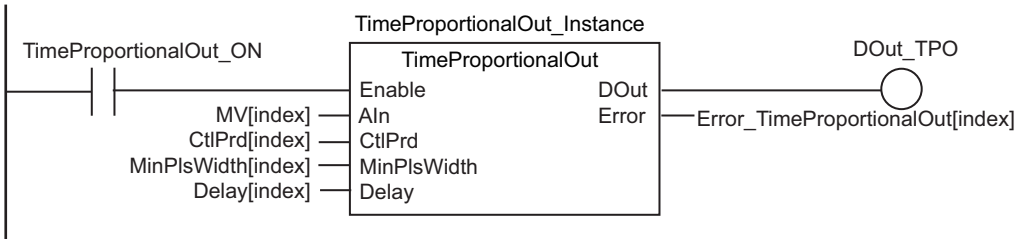
Выдача сигнала для обеспечения безопасности в случае, если возникла ошибка в команде LimitAlarmDvStbySeq_REAL или подан сигнал аварии по выходу за верхний или нижний предел.



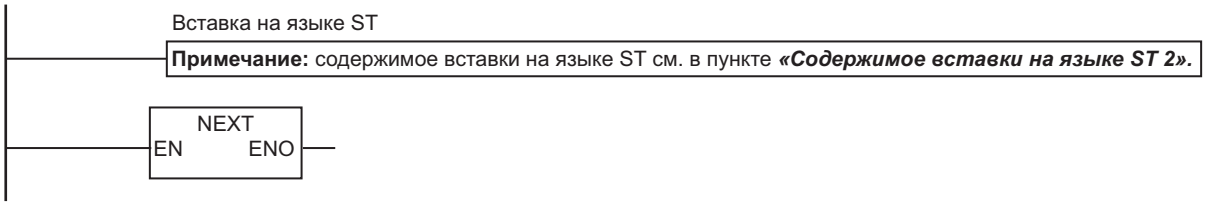
Выполнение команды PIDAT.



Выход широтно-импульсного регулирования



Вывод значений в биты 00…03 выходного слова 1.



● Содержимое вставки на языке ST 1

```
// Получение входных значений 1...4.
CASE index OF
INT#0:
  AI:=AI1;
INT#1:
  AI:=AI2;
INT#2:
  AI:=AI3;
ELSE
  AI:=AI4;
END_CASE;

// Преобразование AI (рег. велич.) в вещественное число.
PV[index]:=INT_TO_REAL(AI)/REAL#10.0;  // CJ1W-PH41U выдает значение, которое в 10
раз больше фактического, поэтому его нужно поделить на 10,0.
```

● Содержимое вставки на языке ST 2

```
// Вывод значений в биты 00…03 выходного слова 1.
CASE index OF
INT#0:
  DO1:=DOut_TPO;
INT#1:
  DO2:=DOut_TPO;
INT#2:
  DO3:=DOut_TPO;
ELSE
  DO4:=DOut_TPO;
END_CASE;
```

Программа на языке ST

Имя	Тип данных	По умолчанию	Сохранение	Комментарий
index	UINT	0	☐	Счетчик цикла
LimitAlarm_ON	BOOL	Истина	☐	Выполнение команды аварии выхода за верхний/нижний предел
LimitAlarmDvStbySeq_ON	BOOL	Истина	☐	Выполнение команды аварии верхнего/нижнего отклонения с контролем последовательности

Имя	Тип данных	По умолчанию	Сохранение	Комментарий
TimeProportionalOut_ON	BOOL	Истина	☐	Выполнение команды широтно-импульсного регулирования
AI	INT	0	☐	Текущее значение
PV	ARRAY[0..3] OF REAL	0,0	☐	Регулируемая величина
SP	ARRAY[0..3] OF REAL	[4(100)]	☐	Заданное значение
DOut_TPO	BOOL	Ложь	☐	Выход широтно-импульсного регулирования
HighVal	ARRAY[0..3] OF REAL	[4(200)]	☐	Установленный верхний предел для сигнализации аварии по выходу за верхний/нижний предел
LowVal	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Установленный нижний предел для сигнализации аварии по выходу за верхний/нижний предел
Hystrs_LimitAlarm	ARRAY[0..3] OF REAL	[4(5)]	☐	Гистерезис аварии по выходу за верхний/нижний предел
Q_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выход сигнала аварии по выходу за верхний/нижний предел
HighAlm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по выходу за верхний предел
LowAlm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по выходу за нижний предел
Error_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде LimitAlarm_REAL
Alm_LimitAlarm	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал для обеспечения безопасности для команды аварии выхода за верхний/нижний предел
DvHighVal	ARRAY[0..3] OF REAL	[4(50)]	☐	Установленное верхнее отклонение для сигнализации аварии по верхнему/нижнему отклонению

Имя	Тип данных	По умолчанию	Сохранение	Комментарий
DvLowVal	ARRAY[0..3] OF REAL	[4(50)]	☐	Установленное нижнее отклонение для сигнализации аварии по верхнему/нижнему отклонению
Q_LimitAlarmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выход сигнала аварии по верхнему/нижнему отклонению
HighAlmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по верхнему отклонению
LowAlmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал аварии по нижнему отклонению
StbySeqFlag	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Флаг контроля последовательности событий
Error_LimitAlarmDvStbySeq	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде LimitAlarmDvStbySeq_REAL
Hystrs_LimitAlarmDv	ARRAY[0..3] OF REAL	[4(3)]	☐	Гистерезис аварии по верхнему/нижнему отклонению
Alm_LimitAlarmDv	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Сигнал для обеспечения безопасности для команды аварии по верхнему/нижнему отклонению
Run	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Условие выполнения
ManCtl	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ручное/автоматическое управление
StartAT	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Условие выполнения автонастройки
OprSetParams	_sOPR_SET_PARAMS	(MVLowLmt:=0.0, MVUpLmt:=100.0, ManResetVal:=0.0, MVTrackSw:=False, MVTrackVal:=0.0, StopMV:=0.0, ErrorMV:=0.0,Alpha:=0.65, ATCalcGain:=1.0, ATHystrs:=0.2)	☐	Рабочие параметры
InitSetParams	_sINIT_SET_PARAMS	(SampTime:=T#100 ms, RngLowLmt:=-10.0,RngUpLmt:=1000.0, DirOpr:=False)	☐	Начальные параметры
PB	ARRAY[0..3] OF REAL	[4(10)]	☑	Зона пропорциональности
TI	ARRAY[0..3] OF TIME	[4(T#0 c)]	☑	Постоянная времени интегрирования
TD	ARRAY[0..3] OF TIME	[4(T#0 c)]	☑	Постоянная времени дифференцирования

Имя	Тип данных	По умолчанию	Сохранение	Комментарий
ManMV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Ручное управляющее воздействие
ATDone	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Нормальное завершение автонастройки
ATBusy	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Выполнение автонастройки
Error_PIDAT	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде PIDAT
ErrorID	ARRAY[0..3] OF WORD	[4(16#0)]	☐	Идентификатор ошибки для команды PIDAT
MV	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Управляющее воздействие
CtlPrd	ARRAY[0..3] OF TIME	[4(T#1 c)]	☐	Период регулирования
MinPlsWidth	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Минимальная ширина импульса
Delay	ARRAY[0..3] OF REAL	[4(0,0)]	☐	Время задержки включения
Error_TimeProportionalOut	ARRAY[0..3] OF BOOL	[4(Ложь)]	☐	Ошибка в команде TimeProportionalOut
LimitAlarm_REAL_instance	LimitAlarm_REAL		☐	
LimitAlarmDvStbySeq_REAL_instance	LimitAlarmDvStbySeq_REAL		☐	
PIDAT_instance	PIDAT		☐	
TimeProportionalOut_instance	TimeProportionalOut		☐	

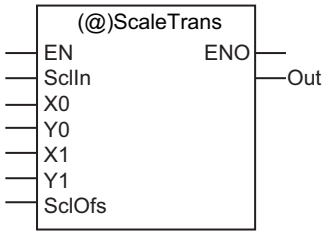
```
// Регулирование температуры в четырех точках.
FOR index:=UINT#0 TO UINT#3 BY UINT#1 DO

  // Получение входных значений 1...4.
  CASE index OF
  INT#0:
    AI:=AI1;
  INT#1:
    AI:=AI2;
  INT#2:
    AI:=AI3;
  ELSE
    AI:=AI4;
  END_CASE;

  // Преобразование AI (рег. велич.) в вещественное число.
```

```
  PV[index]:=INT_TO_REAL(AI)/REAL#10.0; // CJ1W-PH41U выдает значение, которое в 10
 раз больше фактического, поэтому его нужно поделить на 10,0.

  // Авария по выходу за верхний/нижний предел
  LimitAlarm_REAL_instance(
    Enable :=LimitAlarm_ON,
    H :=HighVal[index],
    X :=PV[index],
    L :=LowVal[index],
    EPS :=Hystrs_LimitAlarm[index],
    Q =>Q_LimitAlarm[index],
    QH =>HighAlm[index],
    QL =>LowAlm[index],
    Error =>Error_LimitAlarm[index]);

  // Выдача сигнала для обеспечения безопасности при ошибке в команде LimitAlarm_RE
AL или при аварии по выходу за верхний/нижний предел.
  Alm_LimitAlarm[index]:=Q_LimitAlarm[index] OR Error_LimitAlarm[index];

  // Сигнализация аварии верхнего/нижнего отклонения с контролем
  // последовательности
  LimitAlarmDvStbySeq_REAL_instance(
    Enable :=LimitAlarmDvStbySeq_ON,
    X :=PV[index],
    H :=DvHighVal[index],
    Y :=SP[index],
    L :=DvLowVal[index],
    EPS :=Hystrs_LimitAlarmDv[index],
    Q =>Q_LimitAlarmDv[index],
    QH =>HighAlmDv[index],
    QL =>LowAlmDv[index],
    StbySeqFlag =>StbySeqFlag[index],
    Error =>Error_LimitAlarmDvStbySeq[index]);

  // Выдача сигнала для обеспечения безопасности при ошибке в
 // команде LimitAlarmDvStbySeq_REAL или при аварии по выходу за
 // верхний/нижний предел.
  Alm_LimitAlarmDv[index]:=Q_LimitAlarmDv[index] OR Error_LimitAlarmDvStbySeq[index
];

  // Выполнение команды PIDAT.
  PIDAT_instance(
    Run :=Run[index],
    ManCtl :=ManCtl[index],
    StartAT :=StartAT[index],
    PV :=PV[index],
    SP :=SP[index],
    OprSetParams :=OprSetParams,
    InitSetParams :=InitSetParams,
    ProportionalBand:=PB[index],
    IntegrationTime :=TI[index],
```

```
    DerivativeTime :=TD[index],
    ManMV :=ManMV[index],
    ATDone =>ATDone[index],
    ATBusy =>ATBusy[index],
    Error =>Error_PIDAT[index],
    ErrorID =>ErrorID[index],
    MV =>MV[index]);

  // Выход широтно-импульсного регулирования
  TimeProportionalOut_instance(
    Enable :=TimeProportionalOut_ON,
    AIn :=MV[index],
    CtlPrd :=CtlPrd[index],
    MinPlsWidth :=MinPlsWidth[index],
    Delay :=Delay[index],
    DOut =>DOut_TPO,
    Error =>Error_TimeProportionalOut[index]);

  // Вывод значений в биты 00…03 выходного слова 1.
  CASE index OF
  INT#0:
    DO1:=DOut_TPO;
  INT#1:
    DO2:=DOut_TPO;
  INT#2:
    DO3:=DOut_TPO;
  ELSE
    DO4:=DOut_TPO;
  END_CASE;

END_FOR;
```

ScaleTrans

Команда ScaleTrans преобразует входное значение в выходное значение пропорционально соотношению входного и выходного диапазонов.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ScaleTrans	Изменение масштаба	FUN		Out :=ScaleTrans(ScIIn, X0, Y0, X1, Y1, SclOfs);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.05 или более поздней и Sysmac Studio версии 1.06 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
ScIIn	Входное значение	Вход	Масштабируемое значение	Зависит от типа данных.	---	*1
X0	Нижняя граница входного диапазона		Нижнее предельное значение входного диапазона			0
Y0	Нижняя граница выходного диапазона		Нижнее предельное значение выходного диапазона			
X1	Верхняя граница входного диапазона		Верхнее предельное значение входного диапазона			
Y1	Верхняя граница выходного диапазона		Верхнее предельное значение выходного диапазона			
SclOfs	Смещение		Смещение выходного значения			
Out	Выходное значение	Выход	Значение после изменения масштаба		---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

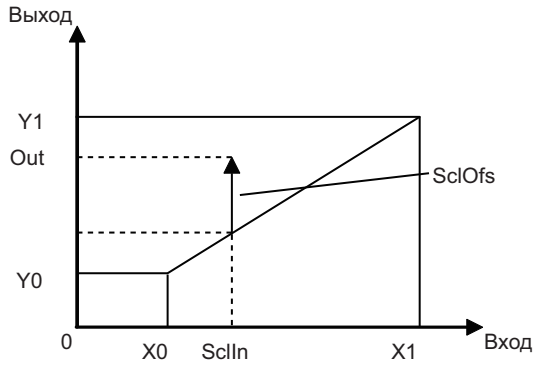
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SclIn														ОК	ОК					
X0														ОК	ОК					
X1														ОК	ОК					
Y0														ОК	ОК					
Y1														ОК	ОК					
SclOfs														ОК	ОК					
Out														ОК	ОК					

Функция

Команда Scale Trans переводит значение *SclIn* из одного масштаба в другой, т. е. пересчитывает входное значение в выходное значение пропорционально соотношению выходного и входного диапазонов значений.
Входной диапазон задается параметрами *X0* (нижняя граница входного диапазона) и *X1* (верхняя граница входного диапазона). Выходной диапазон задается параметрами *Y0* (нижняя граница выходного диапазона) и *Y1* (верхняя граница выходного диапазона).
Значение, приведенное к выходному диапазону, смещается на значение *SclOfs* (смещение), и результат выдается в переменную *Out* (выходное значение). Команду *SclOfs* используют, например, для компенсации ошибки при регулировании температуры.

Преобразование осуществляется по следующей формуле:

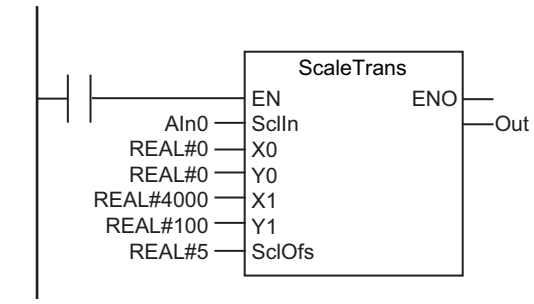
$$Out = \frac{Y1-Y0}{X1-X0}(SclIn - X0) + Y0 + SclOfs$$



Пример записи

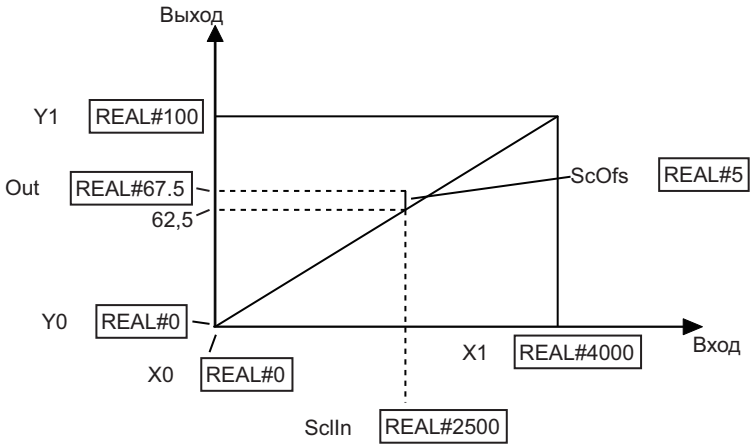
Ниже показан пример применения команды для пересчета входного значения 2500, когда входной диапазон 0...4000 преобразуется в диапазон 0 % ... 100 %. К выходному значению добавляется смещение 5 %.
Используются следующие значения: *SclIn* = REAL#2500, *X0* = REAL#0, *X1* = REAL#4000, *Y0* = REAL#0, *Y1* = REAL#100, а *SclOfs* = REAL#5.
Значение *Out* будет равно REAL#67.5.

LD



ST

```
Out :=ScaleTrans(AIn0,REAL#0,REAL#0,REAL#4000,
                 REAL#100,REAL#5);
```



Входному значению 2500 соответствует значение 62,5 в масштабе 1:40 (входной диапазон 0…4000, выходной диапазон 0…100).
С учетом смещения на 5 значение Out равно REAL#67.5.

Дополнительная информация

- Если команда используется для приведения значения *SclIn* к масштабу значений *PV* или *SP* команды PIDAT, в переменные *Y0* и *Y1* следует передать указанные ниже параметры.

Переменная	Параметр
Y0	*InitSetParams.RngLowLmt* (нижняя граница входного диапазона команды PIDAT)
Y1	*InitSetParams.RngUpLmt* (верхняя граница входного диапазона команды PIDAT)

- Также допускается следующая настройка значений: *X1* < *X0* и *Y1* < *Y0*.

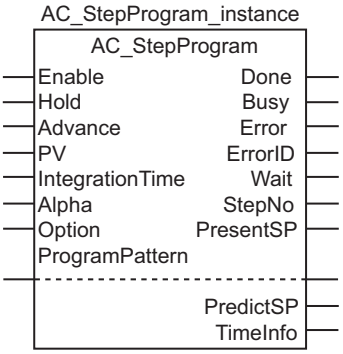
Меры предосторожности для обеспечения надлежащей эксплуатации

Если в *SclIn* передается целочисленный параметр, тип данных преобразуется следующим образом:

Тип данных параметра, передаваемого в *SclIn*	Тип данных *SclIn*, *X0*, *X1*, *Y0*, *Y1* и *SclOfs*
USINT, UINT, SINT или INT	REAL
UDINT или DINT	LREAL
ULINT или LINT	В этом случае произойдет ошибка сборки.

AC_StepProgram

Команда AC_StepProgram производит расчет текущего заданного значения и прогнозируемого заданного значения в каждом периоде выполнения задачи в соответствии с заданной программной последовательностью.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
AC_StepProgram	Поэтапная программа	FB		AC_StepProgram_instance(Enable, Hold, Advance, PV, IntegrationTime, Alpha, Option, ProgramPattern, Done, Busy, Error, ErrorID, Wait, StepNo, PresentSP, PredictSP, TimeInfo);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.06 или более поздней и Sysmac Studio версии 1.07 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Enable	Активация	Вход	ИСТИНА: выполнить ЛОЖЬ: остановить	Зависит от типа данных.	---	ЛОЖЬ
Hold	Приостановка		ИСТИНА: приостановить ЛОЖЬ: не приостанавливать			
Advance	Переход к следующему этапу		Номер выполняемого этапа увеличивается на 1 каждый раз, когда значение этой переменной меняется на ИСТИНА.			
PV	Регулируемая величина		Измеренное значение (регулируемая величина)*1			0
IntegrationTime	Постоянная времени интегрирования		Постоянная времени интегрирования*2	T#0.0000 с ... T#10000.0000 с*3	с	T#0 с
Alpha	Параметр α 2-ПИД регулятора		Параметр α 2-ПИД регулятора*4	0,00...1,00	---	0
Option	Дополнительные параметры		Дополнительные параметры*5	---		---

	Значение	Вход/выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
ProgramPattern[] (массив)	Программная последовательность	Вход-выход	Программная последовательность	---	---	---
Wait	Ожидание	Выход	ИСТИНА: ожидать ЛОЖЬ: не ожидать	Зависит от типа данных.	---	---
StepNo	Номер текущего этапа		Номер текущего этапа	0...255*6		
PresentSP	Текущее заданное значение		Рассчитанное текущее заданное значение	Зависит от типа данных.		
PredictSP	Прогнозируемое заданное значение		Рассчитанное прогнозируемое заданное значение			
TimeInfo	Данные часов		Данные времени для контроля хода выполнения команды	---		

*1. Эта переменная имеет то же назначение, что и переменная *PV* для команды PIDAT. Дополнительные сведения см. в разделе *PV (регулируемая величина)* на стр. 2-900.
*2. Эта переменная имеет то же назначение, что и переменная *IntegrationTime* команды PIDAT. Дополнительные сведения см. в разделе *IntegrationTime (постоянная времени интегрирования)* на стр. 2-900.
*3. Знаки после 0,0001 с отбрасываются.
*4. Эта переменная имеет то же назначение, что и переменная *OprSetParams.Alpha* для команды PIDAT. Дополнительные сведения см. в разделе *Alpha (параметр α 2-ПИД регулятора)* на стр. 2-900.
*5. Дополнительные сведения см. в разделе *Описание структурного типа* на стр. 2-898.
*6. Диапазон допустимых значений для модулей ЦПУ NX701, NX1P2 и модулей ЦПУ серии NJ с версией модуля 1.20 или более ранней, а также для модулей ЦПУ NX102 с версией модуля 1.31 или более ранней: 0...99.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	ОК																			
Hold	ОК																			
Advance	ОК																			
PV														ОК						
IntegrationTime																ОК				
Alpha														ОК						
Option	Подробные сведения о структуре _sAC_STEP_OPTION см. в разделе *Описание структурного типа* на стр. 2-898.																			
ProgramPattern[] (массив)*1*2*3	Подробные сведения о структуре _sAC_STEP_DATA см. в разделе *Описание структурного типа* на стр. 2-898. Следует указать массив.																			
Wait	ОК																			
StepNo						ОК														
PresentSP														ОК						
PredictSP														ОК						

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TimeInfo	Подробные сведения о структуре _sAC_STEP_TIME см. в разделе *Описание структурного типа* на стр. 2-898.																			

*1. Максимальное количество элементов в массиве зависит от версии модуля контроллера или модуля ЦПУ следующим образом:
- Максимальное количество элементов в массиве для модулей ЦПУ NX102 с версией модуля не ниже 1.32, а также модулей ЦПУ NX701, NX1P2 и модулей ЦПУ серии NJ с версией модуля не ниже 1.21: 256.
- Максимальное количество элементов в массиве при любых других версиях модуля, кроме указанных выше: 100.

*2. Это одномерный массив. Если будет указан массив с двумя или большим числом измерений, произойдет ошибка сборки.

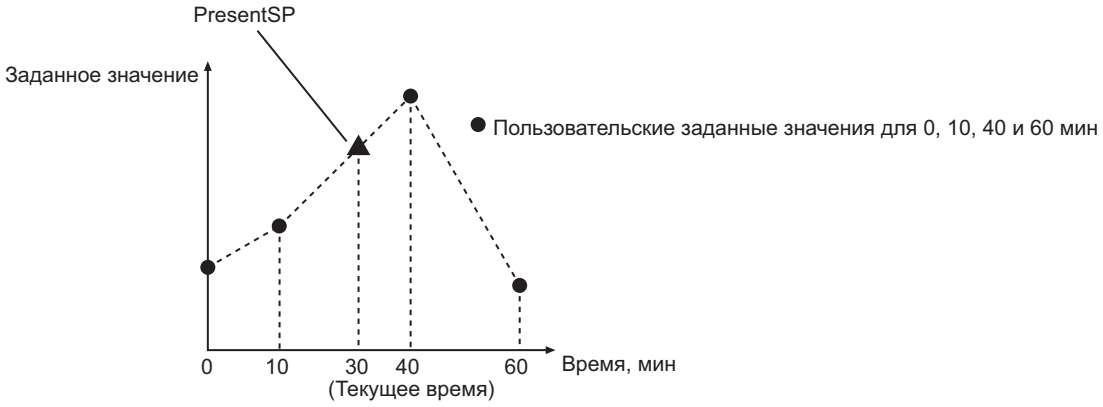
*3. Номер первого элемента массива: 0. Если для первого элемента массива будет указано число, отличное от 0, произойдет ошибка сборки.

Функция

Команда AC_StepProgram производит расчет значений *PresentSP* (текущее заданное значение) и *PredictSP* (прогнозируемое заданное значение) в каждом цикле выполнения задачи с целью формирования соответствующего управляющего воздействия для регулятора температуры при совместном использовании с командой PIDAT.
Текущее заданное значение — это значение уставки в текущем цикле выполнения задачи.
Прогнозируемое заданное значение получается путем применения компенсации задержки 2-ПИД-регулятора к текущему заданному значению.
За счет передачи прогнозируемого заданного значения (*PredictSP*) в качестве заданного значения (*SP*) в команду PIDAT можно улучшить характеристики отслеживания задающего воздействия при программном управлении с использованием команды PIDAT.

PresentSP (текущее заданное значение)

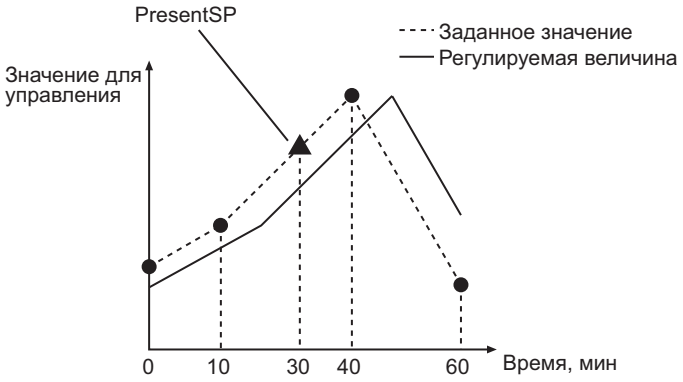
Переменная *PresentSP* (текущее заданное значение) содержит значение уставки в текущем периоде выполнения задачи.
Пусть, например, пользователь задал уставки, которые должны использоваться через 0, 10, 40 и 60 минут после начала регулирования, как показано на рисунке ниже. Предположим также, что после начала регулирования уже прошло 30 минут. Команда AC_StepProgram вычисляет значение *PresentSP* по известным значениям уставок в точках «10 мин» и «40 мин», используя метод линейной интерполяции.



PredictSP (прогнозируемое заданное значение)

Переменная *PredictSP* (прогнозируемое заданное значение) содержит значение уставки, получаемое путем применения компенсации задержки 2-ПИД-регулятора к текущей уставке *PresentSP*.

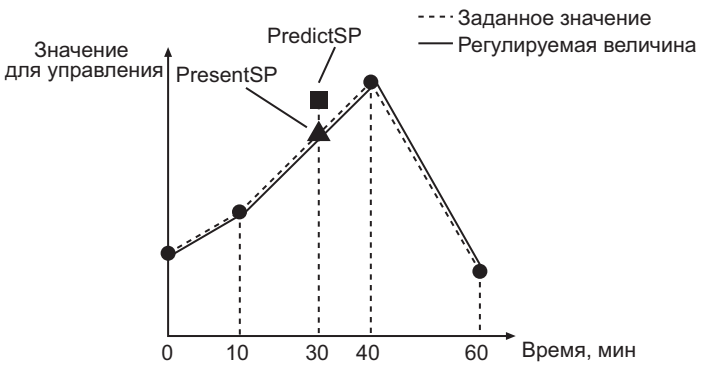
Если значение *PresentSP* будет передаваться в переменную *SP* для команды PIDAT без компенсации задержки, значение *PV*, передаваемое в команду PIDAT, не будет соответствовать заданному значению. Приведенный ниже рисунок поясняет сказанное.



Данную задержку можно компенсировать с помощью переменной *PredictSP*.

Команда AC_StepProgram рассчитывает значение *PredictSP* на основе параметров *IntegrationTime* (постоянная времени интегрирования) и *Alpha* (параметр α 2-ПИД регулятора).

За счет передачи значения *PredictSP* в переменную *SP* команды PIDAT можно улучшить характеристики отслеживания заданного значения при программном управлении с использованием команды PIDAT.



Описание структурного типа

Для переменной *Option* используется структурный тип данных _sAC_STEP_OPTION. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Option	Дополнительные параметры	Дополнительные параметры	_sAC_STEP_OPTION	---		---
StartAtPV	Запуск со значения PV	ИСТИНА: использовать запуск со значения PV ЛОЖЬ: не использовать запуск со значения PV	BOOL	Зависит от типа данных.		ЛОЖЬ
StartStepNo	Номер начального этапа	Номер этапа, с которого следует начать обработку	USINT	0...255*1	---	0
EndStepNo	Номер последнего этапа	Номер этапа, на котором следует завершить обработку*2	USINT			
Reserved	Резерв	Резерв	ARRAY[0..31] OF BYTE	Зависит от типа данных.		Все 32 элемента содержат 0.

*1. Диапазон допустимых значений для модулей ЦПУ NX701, NX1P2 и модулей ЦПУ серии NJ с версией модуля 1.20 или более ранней, а также для модулей ЦПУ NX102 с версией модуля 1.31 или более ранней: 0...99.

*2. Если установлено значение *0*, за номер последнего этапа принимается наибольший номер элемента массива ProgramPattern[].

Для элементов массива ProgramPattern[] (программная последовательность) используется структурный тип данных _sAC_STEP_DATA. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
ProgramPattern	Программная последовательность	Программная последовательность	_sAC_STEP_DATA	---	---	---
ReachSP	Целевое заданное значение	Целевое значение уставки для этапа	REAL	Зависит от типа данных.		0
TimeWidth	Продолжительность	Продолжительность этапа*1	TIME		с	T#0 с
WaitWidth	Ширина ожидания	Ширина ожидания этапа*2	REAL		---	0
WaitTimeLimit	Верхний предел времени ожидания	Предельное время ожидания для заданной ширины ожидания этапа*1*3	TIME		с	T#0 с

*1. Минимальный шаг установки: один период выполнения задачи.

*2. Значение 0 или меньше рассматривается как 0.

*3. Значение 0 или меньше рассматривается как T#0 с.

Для переменной *TimeInfo* (данные времени) используется структурный тип данных _sAC_STEP_TIME. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
TimeInfo	Данные часов	Данные часов	_sAC_STEP_TIME	---	---	---
ProgramTime	Длительность программы	Сумма всех значений *TimeWidth*, с этапа 0 по этап *EndStepNo*.	TIME	Неотрицательное значение	с	T#0 с
ElapseTime	Прошедшее время	Время, прошедшее с момента начала выполнения команды*1	TIME			
ProgressTime	Время выполнения	Время, прошедшее с момента начала выполнения команды*2	TIME			
LeftTime	Оставшееся время	Время, оставшееся до завершения обработки всех этапов на данный момент*2	TIME			
StepProgressTime	Время выполнения этапа	Время, прошедшее с момента начала текущего этапа*2	TIME			
StepLeftTime	Время, оставшееся до завершения этапа	Время, оставшееся до завершения обработки текущего этапа на данный момент*2	TIME			

*1. Включает время ожидания. Не включает время приостановки.
*2. Это значение не включает время ожидания и время приостановки.

Назначение переменных

Ниже поясняется назначение переменных, которые используются в данной команде.

● Enable (активация)

Эта переменная играет роль условия выполнения команды.
Выполнение команды запускается, когда значение *Enable* меняется на ИСТИНА. Когда значение *Enable* меняется на ЛОЖЬ, выполнение команды останавливается.

● Hold (приостановка)

Эта переменная играет роль флага запуска приостановки.
Приостановка выполняется, когда значение *Hold* меняется на ИСТИНА.
Дополнительные сведения о функции приостановки см. в разделе *Приостановка* на стр. 2-906.

● Advance (переход к следующему этапу)

Когда значение этой переменной меняется на ИСТИНА во время выполнения команды, производится переход к следующему этапу обработки.
Дополнительные сведения о смене этапов см. в разделе *Переход к следующему этапу* на стр. 2-908.

- **PV (регулируемая величина)**

  Эта переменная содержит регулируемый технологический параметр управляемой системы. Она имеет то же назначение, что и переменная *PV* для команды PIDAT.

- **IntegrationTime (постоянная времени интегрирования)**

  Эта переменная имеет то же назначение, что и переменная *IntegrationTime* для команды PIDAT. Введите в нее значение переменной *IntegrationTime* для команды PIDAT или команды PIDAT_HeatCool.

- **Alpha (параметр α 2-ПИД регулятора)**

  Эта переменная имеет то же назначение, что и переменная *OprSetParams.Alpha* для команды PIDAT.
  Введите в нее значение переменной *OprSetParams.Alpha* для команды PIDAT или команды PIDAT_HeatCool.

- **StartAtPV (запуск со значения PV)**

  Эта переменная играет роль флага, указывающего, что обработка должна начинаться с текущего значения регулируемой величины.
  Обработка начинается с текущего значения регулируемой величины, когда *StartAtPV* = ИСТИНА.
  Подробные сведения о запуске обработки с текущего значения регулируемой величины см. в разделе *Запуск со значения PV* на стр. 2-906.

- **StartStepNo (номер начального этапа) и EndStepNo (номер последнего этапа)**

  Эти переменные содержат номер этапа программной последовательности, с которого должна начинаться обработка, и номер этапа программной последовательности, которым должна завершаться обработка.
  Если в переменную *EndStepNo* передано значение 0, за номер последнего этапа принимается наибольший номер элемента массива ProgramPattern[].
  Подробные сведения о программных последовательностях и их этапах см. в разделе *Программная последовательность* на стр. 2-902.

- **ReachSP (целевое заданное значение)**

  Эта переменная содержит значение уставки, которое должно быть достигнуто в конце этапа программной последовательности.
  Подробные сведения о программных последовательностях и их этапах см. в разделе *Программная последовательность* на стр. 2-902.

- **TimeWidth (продолжительность)**

  Эта переменная содержит значение длительности этапа программной последовательности.
  Подробные сведения о программных последовательностях и их этапах см. в разделе *Программная последовательность* на стр. 2-902.

- **WaitWidth (ширина ожидания)**

  Эта переменная содержит пороговое значение для перехода в режим ожидания на текущем этапе программной последовательности.
  Дополнительные сведения о режиме ожидания см. в разделе *Ожидание* на стр. 2-904.

● WaitTimeLimit (предельное время ожидания)

Эта переменная содержит значение предельного времени ожидания в режиме ожидания на текущем этапе программной последовательности.
Если значение *WaitTimeLimit* = T#0, время ожидания в режиме ожидания не ограничивается (считается равным бесконечности).
Дополнительные сведения о режиме ожидания см. в разделе *Ожидание* на стр. 2-904.

● Wait (ожидание)

Эта переменная играет роль флага, указывающего, что в данный момент выполняется ожидание.
Если *Wait* = ИСТИНА, значит в данный момент выполняется ожидание.
Дополнительные сведения о режиме ожидания см. в разделе *Ожидание* на стр. 2-904.

● StepNo (номер текущего этапа)

Эта переменная содержит номер текущего этапа.
Подробные сведения о программных последовательностях и их этапах см. в разделе *Программная последовательность* на стр. 2-902.

● PresentSP (текущее заданное значение)

Эта переменная содержит рассчитанное значение текущей уставки.
Дополнительные сведения см. в разделе *PresentSP (текущее заданное значение)* на стр. 2-896.

● PredictSP (прогнозируемое заданное значение)

Эта переменная содержит рассчитанное прогнозируемое значение уставки.
Дополнительные сведения см. в разделе *PredictSP (прогнозируемое заданное значение)* на стр. 2-897.

● ProgramTime (длительность программы)

Эта переменная содержит сумму всех значений *TimeWidth*, с этапа 0 по этап *EndStepNo* программной последовательности.
Подробные сведения о программных последовательностях и их этапах см. в разделе *Программная последовательность* на стр. 2-902.

● ElapseTime (прошедшее время)

Эта переменная содержит значение времени, которое прошло с момента начала выполнения команды. Это значение включает время ожидания, но не включает время приостановки.
Дополнительные сведения о режиме ожидания см. в разделе *Ожидание* на стр. 2-904, а сведения о функции приостановки см. в разделе *Приостановка* на стр. 2-906.

● ProgressTime (время выполнения)

Эта переменная содержит значение времени, которое прошло с момента начала выполнения команды. Это значение не включает время ожидания и время приостановки.
Дополнительные сведения о режиме ожидания см. в разделе *Ожидание* на стр. 2-904, а сведения о функции приостановки см. в разделе *Приостановка* на стр. 2-906.

● LeftTime (оставшееся время)

Эта переменная содержит время, которое на данный момент осталось до завершения обработки всех этапов. Это значение не включает время ожидания и время приостановки.
Дополнительные сведения о режиме ожидания см. в разделе *Ожидание* на стр. 2-904, а сведения о функции приостановки см. в разделе *Приостановка* на стр. 2-906.

● StepProgressTime (время выполнения этапа)

Эта переменная содержит значение времени, которое прошло с начала выполнения текущего этапа программной последовательности. Это значение не включает время ожидания и время приостановки.
Подробные сведения о программных последовательностях и их этапах см. в разделе *Программная последовательность* на стр. 2-902.
Дополнительные сведения о режиме ожидания см. в разделе *Ожидание* на стр. 2-904, а сведения о функции приостановки см. в разделе *Приостановка* на стр. 2-906.

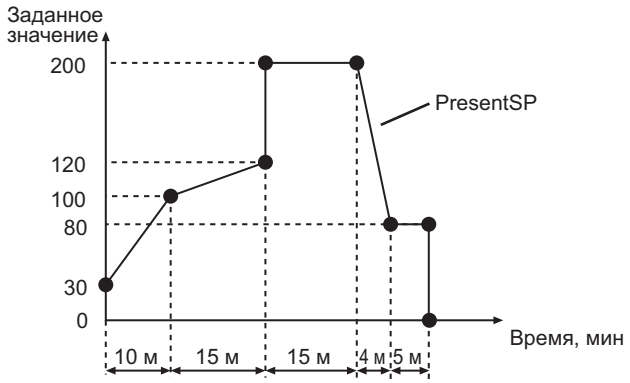
● StepLeftTime (время до завершения этапа)

Эта переменная содержит значение времени, которое на данный момент осталось до завершения обработки текущего этапа программной последовательности. Это значение не включает время ожидания и время приостановки.
Подробные сведения о программных последовательностях и их этапах см. в разделе *Программная последовательность* на стр. 2-902.
Дополнительные сведения о режиме ожидания см. в разделе *Ожидание* на стр. 2-904, а сведения о функции приостановки см. в разделе *Приостановка* на стр. 2-906.

Программная последовательность

Данная команда позволяет запрограммировать изменение уставки ПИД-регулятора во времени. Временной интервал от начала до завершения выполнения команды разбивается на отдельные этапы. Для каждого этапа задаются целевая уставка и продолжительность. Этапы выполняются последовательно, один за другим. Это называется программной последовательностью.
Программная последовательность настраивается с помощью массива ProgramPattern[], содержащего элементы с типом данных _sAC_STEP_DATA. Каждый элемент массива ProgramPattern[] соответствует одному этапу.
Пример программной последовательности приведен ниже. Таблица содержит значения элементов *ReachSP* и *TimeWidth* массива ProgramPattern[], а на рисунке под таблицей показаны значения уставок, соответствующие тому или иному времени, прошедшему с момента выполнения команды.

	Номер элемента ProgramPattern[]							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Номер этапа	0	1	2	3	4	5	6	7
Значение *ReachSP*	30	100	120	200	200	80	80	0
Значение *TimeWidth*	T#0 с	T#10 мин	T#15 мин	T#0 с	T#15 мин	T#4 мин	T#5 мин	T#0 с



На каждом этапе значение уставки (*PresentSP*) в той или иной временной точке рассчитывается на основе известных значений уставок методом линейной интерполяции.

Сплошная линия на рисунке представляет переменную *PresentSP*. В каждом цикле выполнения задачи выдается значение *PresentSP* в соответствующей временной точке.

● Связь между значением *TimeWidth* и длительностью этапа

В следующей таблице отражена взаимосвязь между значением переменной *TimeWidth* и фактической длительностью этапа.

Значение *TimeWidth*	Номер этапа	Продолжительность этапа
T#0 с	0	Принимается равной T#0 с.
	Не 0	Принимается равной одному периоду выполнения задачи.
Положительное	---	Продолжительность этапа равна значению *TimeWidth*.
Отрицательное	---	Принимается равной одному периоду выполнения задачи.

● Работа команды, когда длительность этапа меньше одного периода выполнения задачи

Минимальная единица («разрешение») длительности этапа равна одному периоду выполнению задачи. В следующей таблице показано, как работает команда в случае, когда продолжительность этапа меньше одного периода выполнения задачи.

Номер этапа	Продолжительность этапа	Работа
0	T#0 с	За начальное значение *PresentSP* принимается значение *ReachSP* для этапа 0. Фактическая обработка начинается с этапа 1.
	Не T#0 с	Обработка для текущего этапа выполняется только в течение одного периода выполнения задачи, после чего производится переход к следующему этапу.
Не 0	---	

StartStepNo (номер начального этапа) и *EndStepNo* (номер последнего этапа)

В качестве начального и последнего этапов обработки можно указать любой этап программной последовательности.

Номер начального этапа задается с помощью параметра *StartStepNo*, а номер конечного этапа — с помощью параметра *EndStepNo*.

Например, если задать *StartStepNo* = 3 и *EndStepNo* = 6, то команда выполнит программную последовательность с этапа 3 по этап 6 включительно.

● Изменение значения *StartStepNo* или *EndStepNo* во время выполнения команды

Значения переменных *StartStepNo* и *EndStepNo* можно изменять во время выполнения команды. Ниже показано, как работает команда при изменении этих значений.

Переменная	Новый номер этапа	Работа
StartStepNo	---	Обработка начнется с начала этапа, указанного новым значением *StartStepNo*.
EndStepNo	Новый номер этапа равен или превышает текущей номер этапа	Программная последовательность завершится, когда будет выполнен этап, указанный новым значением *EndStepNo*.
	Новый номер этапа меньше текущего номера этапа	Программная последовательность завершится, как только будет изменен номер последнего этапа. Значение переменной *Done* поменяется на ИСТИНА.

Ожидание

Из-за задержек в управляемой системе значение *PV* может не достигнуть значения *ReachSP* в течение времени, указанного в *TimeWidth* для текущего этапа.
В этом случае можно использовать функцию ожидания для увеличения времени выполнения текущего этапа. То есть длительность этапа будет больше, чем задано в параметре *TimeWidth*.
С функцией ожидания связаны следующие переменные в массиве ProgramPattern[]: *WaitWidth* (ширина ожидания), *WaitTimeLimit* (верхний предел времени ожидания) и *Wait* (ожидание).

● Условия для ожидания

Ожидание выполняется, если разница между *ReachSP* и *PV* превышает значение *WaitWidth* к концу времени выполнения текущего этапа.

● Окончание ожидания

После начала ожидания отсчитывается предельное время ожидания *WaitTimeLimit*. Пока это время не истекло, ожидание заканчивается в момент, когда разница между *ReachSP* и *PV* становится равной или меньшей значения *WaitWidth*. После этого происходит переход к следующему этапу процесса.
Если же разница между *ReachSP* и *PV* так и не станет равной или меньшей значения *WaitWidth* до истечения времени *WaitTimeLimit*, то ожидание окончится, когда истечет время ожидания *WaitTimeLimit*, после чего произойдет переход к следующему этапу процесса. Однако если значение *WaitTimeLimit* = T#0, то верхний предел времени ожидания равен бесконечности. В этом случае время ожидания не ограничивается и ожидание продолжается до тех пор, пока разница между *ReachSP* и *PV* не становится меньшей или равной *WaitWidth*.

● Мониторинг ожидания

Процесс ожидания можно отслеживать по значению переменной *Wait*.
Во время выполнения ожидания значение *Wait* = ИСТИНА.
Если ожидание завершается, значение *Wait* меняется на ЛОЖЬ.

● Переменные времени во время ожидания

В следующей таблице показано, что в режиме ожидания происходит с переменными, связанными с отсчетом времени.

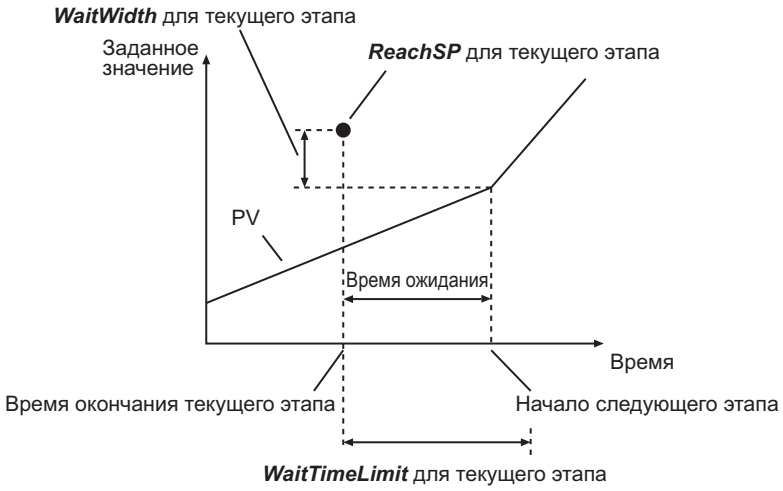
Имя	Работа
ElapseTime	Продолжается отсчет времени.
ProgressTime	Отсчет времени останавливается, сохраняется значение на момент начала ожидания. По окончании ожидания отсчет времени возобновляется с сохраненного значения.
LeftTime	
StepProgressTime	Принимает значение *TimeWidth* для текущего этапа, после чего сохраняет это значение.
StepLeftTime	Принимает значение 0, после чего сохраняет это значение.

● *PresentSP* и *PredictSP* во время ожидания

Во время ожидания в переменных *PresentSP* и *PredictSP* сохраняется значение *ReachSP*.

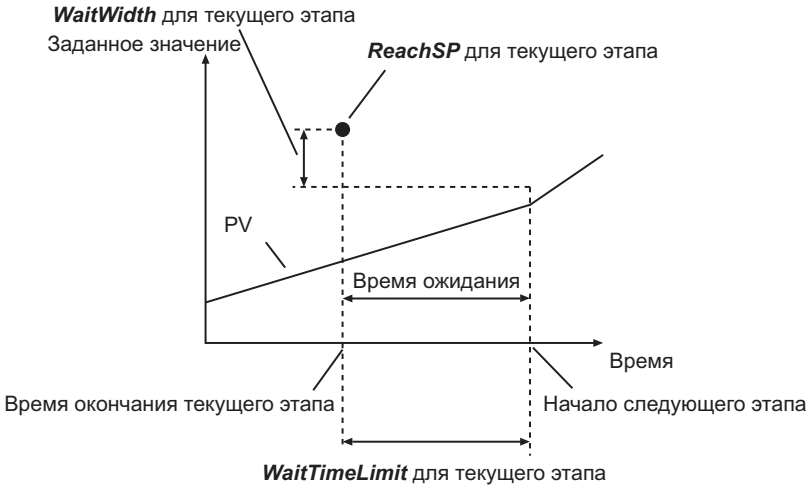
● Пример ожидания

На рисунке ниже показан график изменения значения *PV* для случая, когда разница между *ReachSP* и *PV* становится равной или меньшей значения *WaitWidth* в пределах времени *WaitTimeLimit*.
Разница между *ReachSP* и *PV* превышает значение *WaitWidth* к концу времени выполнения текущего этапа, поэтому выполняется ожидание.
Когда разница между *ReachSP* и *PV* становится меньшей или равной значению *WaitWidth*, происходит переход к следующему этапу процесса.



На рисунке ниже показан график изменения значения *PV* для случая, когда разница между *ReachSP* и *PV* не становится равной или меньшей значения *WaitWidth* за время *WaitTimeLimit*.

Переход к следующему этапу процесса происходит по истечении предельного времени ожидания *WaitTimeLimt*.



Приостановка

Если значение *Hold* = ИСТИНА, обработка для текущего этапа приостанавливается вне зависимости от каких-либо условий. Пока обработка приостановлена, отсчет времени во всех переменных, связанных с отсчетом времени, также остановлен.
Отсчет времени в переменных, связанных с отсчетом времени, возобновляется, когда значение *Hold* меняется на ЛОЖЬ.

● Отсчет времени во время приостановки

В следующей таблице показано, что происходит с переменными, связанными с отсчетом времени, когда обработка приостановлена.

Имя	Работа
ElapseTime	Отсчет времени останавливается, сохраняется значение на момент начала приостановки. По окончании приостановки отсчет времени возобновляется с сохраненного значения.
ProgressTime	
LeftTime	
StepProgressTime	
StepLeftTime	

● *PresentSP* и *PredictSP* во время приостановки

Пока обработка приостановлена, *PresentSP* сохраняет значение на момент начала приостановки.
Пока обработка приостановлена, *PredictSP* содержит такое же значение, что и *PresentSP*.

● Приостановка во время ожидания

Если обработка приостанавливается во время ожидания, ожидание завершается. Поэтому значение переменной *Wait* меняется на ЛОЖЬ. По окончании приостановки снова оцениваются условия для ожидания.

Запуск со значения PV

Обработку также можно начинать в точке программной последовательности, в которой значение *PresentSP* равно текущему значению *PV* на момент активации команды.

Если значение *StartAtPV* = ИСТИНА в момент перехода *Enable* в состояние ИСТИНА, используется запуск со значения PV.
В случае запуска программной последовательности с текущего значения PV соблюдается следующий порядок действий.

1 Считывается значение *PV*.

2 В пределах последовательности (с этапа 0 по последний этап) производится поиск временной точки, в которой значение *PresentSP* в первый раз становится равно значению *PV*.
Если значение *PresentSP* возрастает, начиная с этапа 0, поиск выполняется до тех пор, пока значение *PresentSP* не начинает уменьшаться. Точно так же, если значение *PresentSP* уменьшается, начиная с этапа 0, поиск выполняется до тех пор, пока значение *PresentSP* не начинает возрастать.

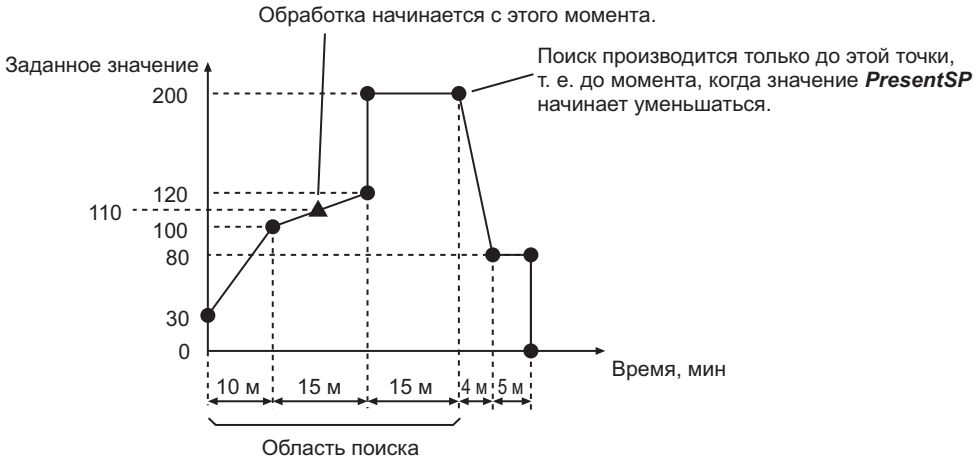
3 Обработка начинается в точке, найденной описанным выше образом.

Если в области поиска не будет обнаружена временная точка, в которой *PV* и *PresentSP* имеют одинаковое значение, обработка будет начата с этапа 0.

Пример запуска обработки с текущего значения PV показан ниже. Элементы массива ProgramPattern[] содержат следующие значения.

	Номер элемента ProgramPattern[]							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Номер этапа	0	1	2	3	4	5	6	7
Значение *ReachSP*	30	100	120	200	200	80	80	0
Значение *TimeWidth*	T#0 с	T#10 мин	T#15 мин	T#0 с	T#15 мин	T#4 мин	T#5 мин	T#0 с

В данном примере значение *PresentSP* возрастает, начиная с этапа 0. Поэтому поиск производится только в интервале 40 минут после начала обработки, т. е. до точки, в которой значение *PresentSP* начинает уменьшаться.
Предположим, что значение *PV* в момент запуска выполнения команды было равно 110. Как видно из рисунка ниже, обработка в этом случае начинается в точке, в которой значение *PresentSP* равно 110.



● Отсчет времени при запуске со значения PV

В следующей таблице показано, что происходит с переменными, связанными с отсчетом времени, при запуске обработки со значения PV.

Имя	Работа
ElapseTime	Содержит 0.
ProgressTime	Указывает время от этапа 0 до точки, найденной при поиске.
LeftTime	Указывает время от настоящего момента до конца этапа *EndStepNo*.
StepProgressTime	Указывает время от начала текущего этапа до точки, найденной при поиске.
StepLeftTime	Указывает время от настоящего момента до полного завершения обработки текущего этапа.

● Изменение значения *StartAtPV* во время выполнения команды

Любые изменения в значении *StartAtPV* во время выполнения команды игнорируются.

Переход к следующему этапу

Если значение переменной *Advance* меняется на ИСТИНА во время выполнения команды, производится переход к началу следующего этапа программной последовательности.

● Отсчет времени при переходе к следующему этапу

В следующей таблице показано, как работают переменные, связанные с отсчетом времени, когда производится переход к следующему этапу программной последовательности.

Имя	Работа
ElapseTime	Продолжается отсчет времени.
ProgressTime	Содержит сумму значений *TimeWidth* с этапа 0 по текущий этап.
LeftTime	Указывает время от следующего этапа до конца этапа *EndStepNo*.
StepProgressTime	Содержит 0, так как производится переход в начало следующего этапа.
StepLeftTime	Содержит значение *TimeWidth* для следующего этапа.

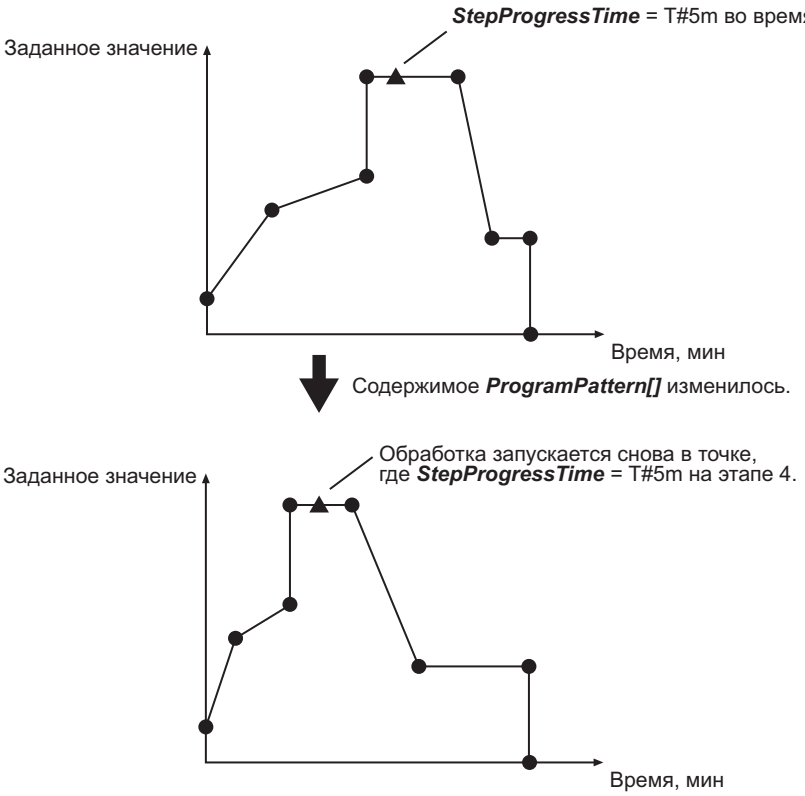
● Изменение значения *StartStepNo* одновременно с переходом к следующему этапу

Если значение переменной *StartStepNo* изменяется одновременно с переходом переменной *Advance* в состояние ИСТИНА, приоритетом обладает изменение значения переменной *StartStepNo*. Поэтому в этом случае производится переход к началу этапа *StartStepNo*.

Изменение программной последовательности во время выполнения команды

Содержимое массива ProgramPattern[] можно изменять во время выполнения команды.
При изменении содержимого массива ProgramPattern[] значение *PresentSP* рассчитывается снова.
Обработка перезапускается во временной точке *StepProgressTime* этапа, который выполнялся непосредственно перед изменением программной последовательности.
Также можно изменить содержимое для предыдущих этапов.
Пусть, например, содержимое массива ProgramPattern[] изменяется во время выполнения этапа 4.

Предположим также, что предыдущее значение *StepProgressTime* было равно T#5 мин. После изменения программной последовательности обработка запустится снова в точке T#5 мин (значение *StepProgressTime*) этапа 4.



Если значение *TimeWidth* для этапа меньше, чем значение *StepProgressTime*, обработка возобновится с начала следующего этапа.

● Отсчет времени при изменении программной последовательности во время выполнения команды

В следующей таблице показано, как работают переменные, связанные с отсчетом времени, когда производится изменение программной последовательности во время выполнения команды.

Имя	Работа
ProgramTime	Содержит сумму значений *TimeWidth* с этапа 0 по этап *EndStepNo* после изменения.
ElapseTime	Продолжается отсчет времени.
ProgressTime	Содержит сумму всех значений *StepProgressTime* и всех значений *TimeWidth* с этапа 0 по этап, предшествующий текущему этапу, после изменения.
LeftTime	Указывает время от настоящего момента до конца этапа *EndStepNo* после изменения.
StepProgressTime	Отсчет времени продолжается со значения, которое было перед изменением.
StepLeftTime	Указывает время от настоящего момента текущего этапа до полного завершения обработки текущего этапа после изменения.

● Изменение программной последовательности во время ожидания

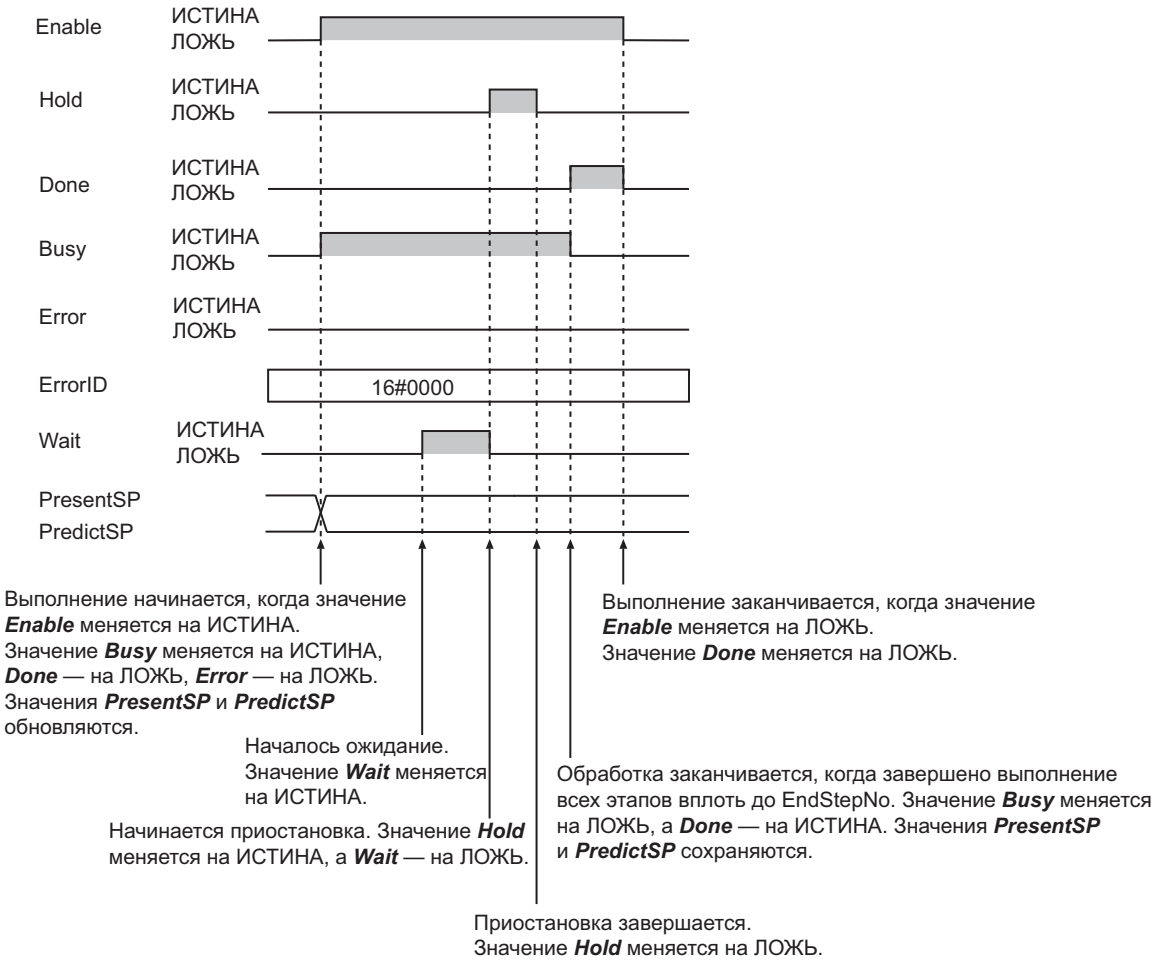
Если программная последовательность будет изменена во время ожидания, оценка условий ожидания будет выполнена вновь для пересчитанного значения *PresentSP*.
Однако если после изменения значение *StepProgressTime* больше, чем значение *WaitTimeLimit*, ожидание немедленно прекращается и происходит переход к следующему этапу последовательности.

● Изменение программной последовательности во время приостановки

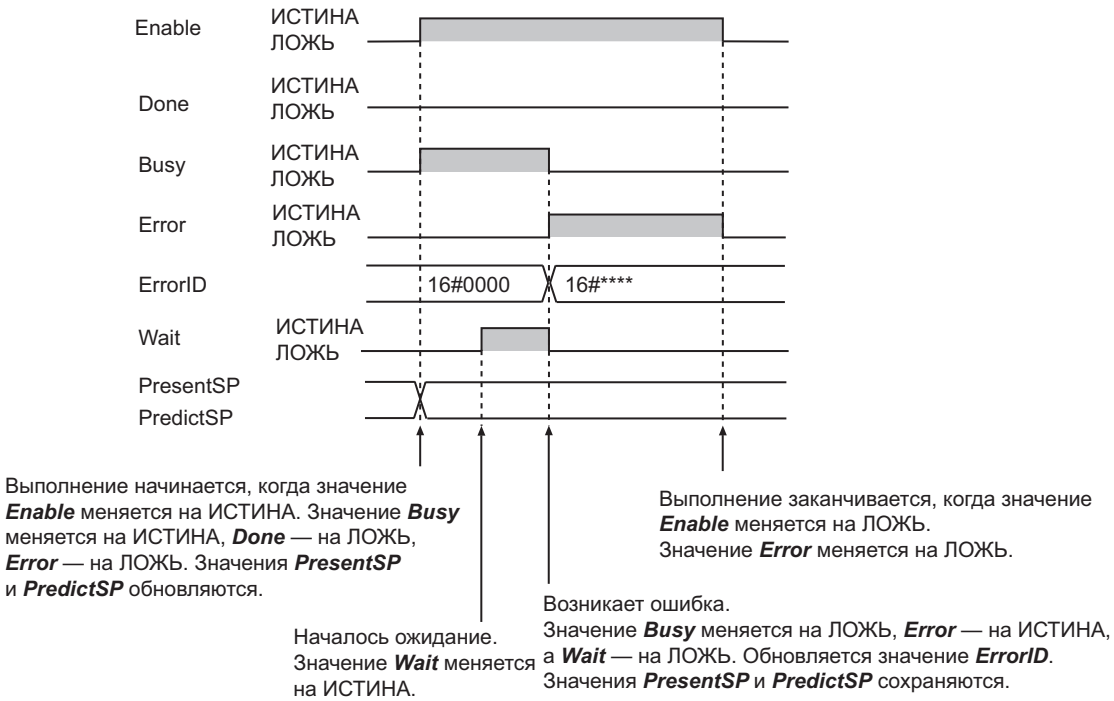
Если программная последовательность будет изменена во время приостановки, приостановка будет продолжена для пересчитанного значения *PresentSP*.

Временные диаграммы

На следующем рисунке показана временная диаграмма в случае работы без ошибок.



На следующем рисунке показана временная диаграмма для случая, когда возникает ошибка.



Меры предосторожности для обеспечения надлежащей эксплуатации

В указанных ниже случаях произойдет ошибка. Выход *Error* перейдет в состояние ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки.

Ошибка	Значение *ErrorID*
Значение переменной *IntegrationTime*, *Alpha*, *StartStepNo* или *EndStepNo* выходит за допустимый диапазон.	16#0400
Номер последнего элемента в массиве ProgramPattern[] превышает 255*1.	16#0416

*1. Номер последнего элемента для модулей ЦПУ NX701, NX1P2 и модуля ЦПУ серии NJ с версией модуля 1.20 или более ранней, а также для модуля ЦПУ NX102 с версией модуля 1.31 или более ранней: 99.

Пример программы

В качестве примера рассмотрим программу, которая выполняет регулирование температуры с оптимальными значениями параметров ПИД-контура для каждого этапа команды AC_StepProgram.

Порядок выполнения операций

Рассматриваемая программа выполняет две операции:

- Рассчитывает оптимальные параметры ПИД-регулирования для каждого этапа.
- Выполняет регулирование температуры в соответствии с заданной программной последовательностью.

Каждая из этих операций подробно рассматривается ниже.

● Расчет оптимальных параметров ПИД-регулятора для каждого этапа

До начала регулирования температуры в соответствии с программной последовательностью необходимо произвести расчет оптимальных ПИД-констант для каждого этапа. Для расчета оптимальных ПИД-констант используется функция автонастройки команды PIDAT.
Рассчитанные значения ПИД-констант сохраняются в массив структур PIDbank[], а в качестве индексов элементов выступают номера этапов программной последовательности. Каждый элемент массива PIDbank[] представляет собой структуру, члены которой содержат значения зоны пропорциональности, постоянной времени интегрирования и постоянной времени дифференцирования для соответствующего этапа регулирования.

Соблюдается следующий порядок действий:

1 Значение переменной *ACSP_Enable* команды AC_StepProgram меняется на ИСТИНА.
Начинается выполнение команды AC_StepProgram, и значение переменной *StepNo* (номер текущего этапа) меняется на 0.

2 Значение переменной *Run* команды PIDAT меняется на ИСТИНА.
Выполняется команда PIDAT.

3 Значение переменной *StartAT* (условие выполнения автонастройки) меняется на ИСТИНА.
Значение переменной *Hold* команды AC_StepProgram меняется на ИСТИНА, и программная последовательность приостанавливается.
Выполняется автонастройка в рамках команды PIDAT, и рассчитываются оптимальные параметры ПИД-регулирования для этапа 0.

4 Автонастройка завершается.
Значение переменной *ATDone* (нормальное завершение автонастройки) команды PIDAT меняется на ИСТИНА.
Рассчитанные ПИД-константы сохраняются в элемент PIDbank[0].

5 Значение переменной *Hold* команды AC_StepProgram меняется на ЛОЖЬ.
Приостановка выполнения команды AC_StepProgram отменяется.
Через некоторое время происходит переход к следующему этапу, и значение *StepNo* меняется на 1.

6 Для каждого этапа повторяются шаги с 3 по 5.
В результате в массив PIDbank[] будут сохранены оптимальные значения параметров ПИД-регулирования для всех этапов.

● Регулирование температуры в соответствии с программной последовательностью

Для регулирования температуры в соответствии с программной последовательностью используются оптимальные значения постоянных ПИД-регулирования для каждого этапа.

Соблюдается следующий порядок действий:

1 Значение переменной *ACSP_Enable* команды AC_StepProgram меняется на ИСТИНА.
Начинается выполнение команды AC_StepProgram, и значение переменной *StepNo* (номер этапа) меняется на 0.

2 Значение переменной *Run* команды PIDAT меняется на ИСТИНА.

Выполняется команда PIDAT.

3 В каждом цикле выполнения задачи на выход команды PIDAT подается управляющее воздействие *MV*.

4 Команда TimeProportionalOut формирует выходной сигнал широтно-импульсного регулирования в соответствии со значением *MV*.

5 Через некоторое время происходит переход к следующему этапу.

6 Шаги 3...5 повторяются, пока не будет достигнут и выполнен последний этап программной последовательности.

Настройка параметров в Sysmac Studio

Для использования рассматриваемой программы необходимо использовать Sysmac Studio для настройки конфигурации сети, карты входов-выходов и определений типов данных.

● Параметры сети

Конфигурация сети приведена в следующей таблице. Ведомый терминал с указанной ниже конфигурацией подключен к узлу сети EtherCAT с адресом узла 1. В программе используются имена устройств, указанные в таблице.

Номер модуля	Номер модели	Модуль	Имя устройства
0	NX-ECC201	Интерфейсный модуль EtherCAT	E001
1	NX-TS2101	Модуль температурных входов	N1
2	NX-OD3121	Модуль дискретных выходов	N2

● Карта входов-выходов

В таблице ниже перечислены используемые параметры карты соответствия входов и выходов (I/O map).

Позиция	Порт	Описание	Чтение/запись	Тип данных	Переменная	Тип переменной
Unit1	Канал 1, измеренное значение, тип REAL*1	Измеренное значение канала (REAL)	Чт.	REAL	N1_Ch1_Measured_Value_REAL	Глобальная переменная
Unit1	Канал 2, измеренное значение, тип REAL*2	Измеренное значение канала (REAL)	Чт.	REAL	N1_Ch2_Measured_Value_REAL	Глобальная переменная
Unit2	Выходной бит 00	Выходной бит 00	Зап.	BOOL	N2_Output_Bit_00	Глобальная переменная
Unit2	Выходной бит 01	Выходной бит 01	Зап.	BOOL	N2_Output_Bit_01	Глобальная переменная
Unit2	Выходной бит 02	Выходной бит 02	Зап.	BOOL	N2_Output_Bit_02	Глобальная переменная

Позиция	Порт	Описание	Чтение/запись	Тип данных	Переменная	Тип переменной
Unit2	Выходной бит 03	Выходной бит 03	Зап.	BOOL	N2_Output_Bit_03	Глобальная переменная

*1. К записям ввода-вывода для модуля температурных входов NX-TS2101 необходимо добавлять 0x6003:01 (канал 1, измеренное значение, тип REAL).

*2. К записям ввода-вывода для модуля температурных входов NX-TS2101 необходимо добавлять 0x6003:02 (канал 2, измеренное значение, тип REAL).

● Определения типов данных

В следующей таблице приведено определение структуры sPID_BANK.

Структура	Имя	Тип данных	Комментарий
▼	sPID_BANK	STRUCT	Структура параметров ПИД-регулятора
	PB	REAL	Зона пропорциональности
	TI	TIME	Постоянная времени интегрирования
	TD	TIME	Постоянная времени дифференцирования

Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	ACSP_Enable	BOOL	ЛОЖЬ	Активация команды AC_StepProgram
	Hold	BOOL	ЛОЖЬ	Приостановка
	Advance	BOOL	ЛОЖЬ	Переход к следующему этапу
	Option	_sAC_STEP_OPTION	(StartAtPV:=FALSE, StartStepNo:=0, EndStepNo:=7, Reserved:=[32(16#0)])	Дополнительные параметры

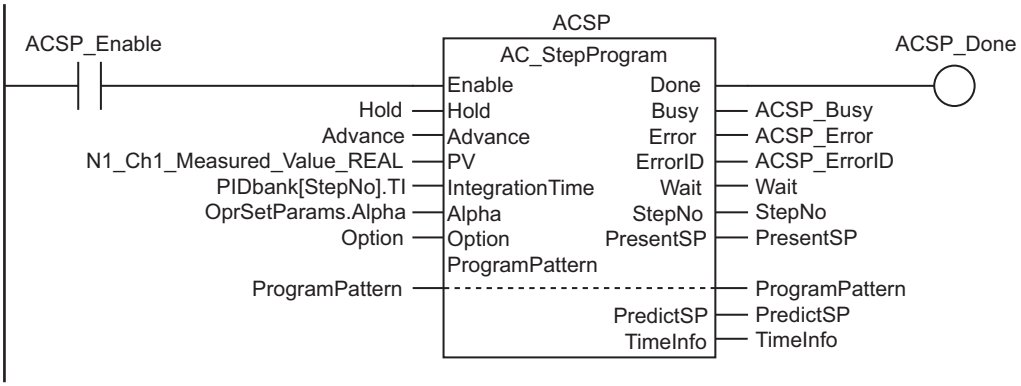
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	ProgramPattern	ARRAY[0..7] OF _sAC_STEP_DATA	[(ReachSP:=30.0, TimeWidth:=T#0 s, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=100.0, TimeWidth:=T#10 m, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=120.0, TimeWidth:=T#15 m, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=150.0, TimeWidth:=T#0 s, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=150.0, TimeWidth:=T#15 m, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=80.0, TimeWidth:=T#4 m, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=80.0, TimeWidth:=T#5 m, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=10.0, TimeWidth:=T#0 s, WaitWidth:=3.0, WaitTimeLimit:=T#1 m)]	Программная последовательность
	ACSP_Busy	BOOL	ЛОЖЬ	Идет выполнение команды AC_StepProgram
	ACSP_Error	BOOL	ЛОЖЬ	Ошибка команды AC_StepProgram
	ACSP_ErrorID	WORD	WORD#16#0	Код ошибки команды AC_StepProgram
	Wait	BOOL	ЛОЖЬ	Ожидание
	StepNo	USINT	0	Номер текущего этапа
	PresentSP	REAL	0,0	Текущее заданное значение
	PredictSP	REAL	0,0	Прогнозируемое заданное значение

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	TimeInfo	_sAC_STEP_TIME	(ProgramTime:=T#0 s, ElapseTime:=T#0 s, ProgressTime:=T#0 s, LeftTime:=T#0 s, StepProgressTime:=T#0 s, StepLeftTime:=T#0 s)	Данные часов
	ACSP_Done	BOOL	ЛОЖЬ	Завершение команды AC_StepProgram
	Run	BOOL	ЛОЖЬ	Условие выполнения команды PIDAT
	ManCtl	BOOL	ЛОЖЬ	Ручное/автоматическое управление
	StartAT	BOOL	ЛОЖЬ	Условие выполнения автонастройки
	OprSetParams	_sOPR_SET_PARAMS	(MVLowLmt:=0.0, MVUpLmt:=100.0, ManResetVal:=0.0, MVTrackSw:=FALSE, MVTrackVal:=0.0, StopMV:=0.0, ErrorMV:=0.0, Alpha:=0.65, ATCalcGain:=1.0, ATHystrs:=0.2)	Рабочие параметры
	InitSetParams	_sINIT_SET_PARAMS	(SampTime:=T#250 ms, RngLowLmt:=-200.0, RngUpLmt:=1300.0, DirOpr:=FALSE)	Начальные параметры
	ManMV	REAL	0,0	Ручное управляющее воздействие
	ATBusy	BOOL	ЛОЖЬ	Выполнение автонастройки
	PID_ErrorID	WORD	WORD#16#0	Код ошибки команды PIDAT
	PID_Error	BOOL	ЛОЖЬ	Ошибка команды PIDAT
	MV	REAL	0,0	Управляющее воздействие
	ATDone	BOOL	ЛОЖЬ	Нормальное завершение автонастройки
	TPO_Error	BOOL	ЛОЖЬ	Ошибка команды TimeProportionalOut
	PIDbank	ARRAY[0..7] OF sPID_BANK	[8((PB:=10, TI:=T#233 s, TD:=T#60 s))]	Массив хранения оптимальных ПИД-параметров
	ACSP	AC_StepProgram		
	PID	PIDAT		
	TPO	TimeProportionalOut		

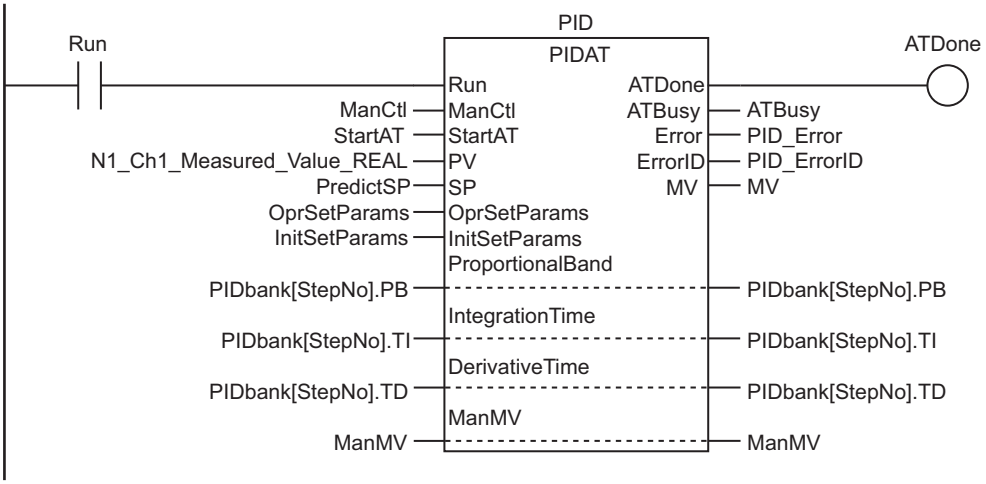
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	N1_Ch1_Measured_Value_REAL	REAL	☐	Измеренное значение канала (REAL)
	N2_Output_Bit_00	BOOL	☐	Бит выхода

Выполнение приостановки для команды AC_StepProgram во время автонастройки.

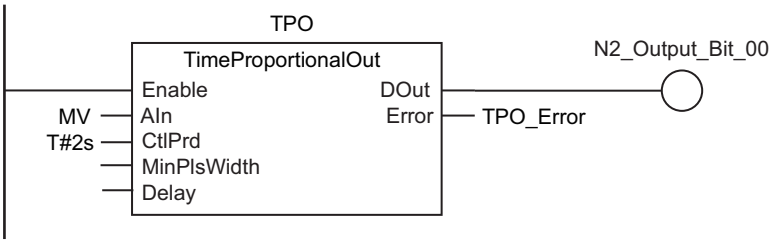
Выполнение команды AC_StepProgram.



Выполнение команды PIDAT.



Выполнение команды TimeProportionalOut.



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	ACSP_Enable	BOOL	ЛОЖЬ	Активация команды AC_StepProgram
	Hold	BOOL	ЛОЖЬ	Приостановка
	Advance	BOOL	ЛОЖЬ	Переход к следующему этапу
	Option	_sAC_STEP_OPTION	(StartAtPV:=FALSE, StartStepNo:=0, EndStepNo:=7, Reserved:=[32(16#0)])	Дополнительные параметры
	ProgramPattern	ARRAY[0..7] OF _sAC_STEP_DATA	[(ReachSP:=30.0, TimeWidth:=T#0 s, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=100.0, TimeWidth:=T#10 m, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=120.0, TimeWidth:=T#15 m, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=150.0, TimeWidth:=T#0 s, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=150.0, TimeWidth:=T#15 m, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=80.0, TimeWidth:=T#4 m, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=80.0, TimeWidth:=T#5 m, WaitWidth:=3.0, WaitTimeLimit:=T#1 m), (ReachSP:=10.0, TimeWidth:=T#0 s, WaitWidth:=3.0, WaitTimeLimit:=T#1 m)]	Программная последовательность
	ACSP_Busy	BOOL	ЛОЖЬ	Идет выполнение команды AC_StepProgram
	ACSP_Error	BOOL	ЛОЖЬ	Ошибка команды AC_StepProgram

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	ACSP_ErrorID	WORD	WORD#16#0	Код ошибки команды AC_StepProgram
	Wait	BOOL	ЛОЖЬ	Ожидание
	StepNo	USINT	0	Номер текущего этапа
	PresentSP	REAL	0,0	Текущее заданное значение
	PredictSP	REAL	0,0	Прогнозируемое заданное значение
	TimeInfo	_sAC_STEP_TIME	(ProgramTime:=T#0 s, ElapseTime:=T#0 s, ProgressTime:=T#0 s, LeftTime:=T#0 s, StepProgressTime:=T#0 s, StepLeftTime:=T#0 s)	Данные часов
	ACSP_Done	BOOL	ЛОЖЬ	Завершение команды AC_StepProgram
	Run	BOOL	ЛОЖЬ	Условие выполнения команды PIDAT
	ManCtl	BOOL	ЛОЖЬ	Ручное/автоматическое управление
	StartAT	BOOL	ЛОЖЬ	Условие выполнения автонастройки
	PreStartAT	BOOL	ИСТИНА	Условие выполнения автонастройки для предыдущего цикла выполнения задачи
	OprSetParams	_sOPR_SET_PARAMS	(MVLowLmt:=0.0, MVUpLmt:=100.0, ManResetVal:=0.0, MVTrackSw:=FALSE, MVTrackVal:=0.0, StopMV:=0.0, ErrorMV:=0.0, Alpha:=0.65, ATCalcGain:=1.0, ATHystrs:=0.2)	Рабочие параметры
	InitSetParams	_sINIT_SET_PARAMS	(SampTime:=T#250 ms, RngLowLmt:=-200.0, RngUpLmt:=1300.0, DirOpr:=FALSE)	Начальные параметры
	ManMV	REAL	0,0	Ручное управляющее воздействие
	ATBusy	BOOL	ЛОЖЬ	Выполнение автонастройки
	PID_ErrorID	WORD	WORD#16#0	Код ошибки команды PIDAT
	PID_Error	BOOL	ЛОЖЬ	Ошибка команды PIDAT

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	MV	REAL	0,0	Управляющее воздействие
	ATDone	BOOL	ЛОЖЬ	Нормальное завершение автонастройки
	TPO_Error	BOOL	ЛОЖЬ	Ошибка команды TimeProportionalOut
	PIDbank	ARRAY[0..7] OF sPID_BANK	[8((PB:=10, TI:=T#233 s, TD:=T#60 s))]	Массив хранения оптимальных ПИД-параметров
	TPO_Enable	BOOL	ЛОЖЬ	Активация команды TimeProportionalOut
	MinPlsWidth	REAL	0,0	Минимальная ширина импульса
	Delay	REAL	0,0	Задержка
	ACSP	AC_StepProgram		
	PID	PIDAT		
	TPO	TimeProportionalOut		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	N1_Ch1_Measured_Value_REAL	REAL	☐	Измеренное значение канала (REAL)
	N2_Output_Bit_00	BOOL	☐	Бит выхода

```
TPO_Enable := TRUE;

// Приостановка команды AC_StepProgram во время автонастройки.
IF StartAT AND PreStartAT=FALSE THEN
  Hold := TRUE;
END_IF;
PreStartAT := StartAT;

// Выполнение команды AC_StepProgram.
IF ACSP_Enable THEN
  ACSP(Enable :=ACSP_Enable,
    Hold :=Hold,
    Advance :=Advance,
    PV :=N1_Ch1_Measured_Value_REAL,
    IntegrationTime:=PIDbank[StepNo].TI,
    Alpha :=OprSetParams.Alpha,
    Option :=Option,
    ProgramPattern :=ProgramPattern,
```

```
    Done =>ACSP_Done,
    Busy =>ACSP_Busy,
    Error =>ACSP_Error,
    ErrorID =>ACSP_ErrorID,
    Wait =>Wait,
    StepNo =>StepNo,
    PresentSP =>PresentSP,
    PredictSP =>PredictSP,
    TimeInfo =>TimeInfo);
END_IF;

// Выполнение команды PIDAT.
IF Run THEN
  PID(Run :=Run,
    ManCtl :=ManCtl,
    StartAT :=StartAT,
    PV :=N1_Ch1_Measured_Value_REAL,
    SP :=PredictSP,
    OprSetParams :=OprSetParams,
    InitSetParams :=InitSetParams,
    ProportionalBand:=PIDbank[StepNo].PB,
    IntegrationTime :=PIDbank[StepNo].TI,
    DerivativeTime :=PIDbank[StepNo].TD,
    ManMV :=ManMV,
    ATDone =>ATDone,
    ATBusy =>ATBusy,
    Error =>PID_Error,
    ErrorID =>PID_ErrorID,
    MV=>MV);
END_IF;

// Выполнение команды TimeProportionalOut.
TPO(Enable :=TPO_Enable,
  AIn :=MV,
  CtlPrd :=T#2s,
  MinPlsWidth:=MinPlsWidth,
  Delay :=Delay,
  DOut =>N2_Output_Bit_00,
  Error =>TPO_Error);
```

Команды для управления системой

Команда	Имя	Стр.
TraceSamp	Отбор данных для протокола данных	стр. 2-925
TraceTrig	Запуск протокола данных	стр. 2-929
GetTraceStatus	Чтение состояния протокола данных	стр. 2-932
SetAlarm	Создание ошибки пользователя	стр. 2-936
ResetAlarm	Сброс ошибки пользователя	стр. 2-941
GetAlarm	Получить состояние ошибки пользователя	стр. 2-943
ResetPLCError	Сброс ошибки контроллера в PLC	стр. 2-945
GetPLCError	Получить состояние ошибки контроллера в PLC	стр. 2-949
ResetCJBError	Сброс ошибки контроллера в шине CJ	стр. 2-951
GetCJBError	Получить состояние ошибки шины ввода-вывода	стр. 2-953
GetEIPError	Получить состояние ошибки в EtherNet/IP	стр. 2-955
ResetMCError	Сброс ошибки в Motion Control	стр. 2-957
GetMCError	Получить состояние ошибки в Motion Control	стр. 2-963
ResetECError	Сброс ошибки в EtherCAT	стр. 2-966
GetECError	Получить состояние ошибки в EtherCAT	стр. 2-968
ResetNXBError	Сброс ошибки в NX Bus	стр. 2-971
GetNXBError	Получить состояние ошибки в NX Bus	стр. 2-973
GetNXUnitError	Получить состояние ошибки в модуле NX	стр. 2-975
SetInfo	Создание информации пользователя	стр. 2-983
ResetUnit	Перезапустить модуль	стр. 2-985
GetNTPStatus	Чтение состояния NTP	стр. 2-990
RestartNXUnit	Перезапуск модуля NX	стр. 2-992
NX_ChangeWriteMode	Перевод модуля NX в режим записи	стр. 2-998

Команда	Имя	Стр.
NX_SaveParam	Сохранение параметров модуля NX	стр. 2-1004
PLC_ReadTotalPowerOnTime	Чтение общего времени работы ПЛК	стр. 2-1010
NX_ReadTotalPowerOnTime	Чтение общего времени работы модуля NX	стр. 2-1013

TraceSamp

Команда TraceSamp выполняет отбор данных для протокола данных.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TraceSamp	Отбор данных для протокола данных	FUN	(@)TraceSamp EN ENO TraceNo Out Point	TraceSamp(TraceNo, Point);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
TraceNo	Номер протокола	Вход	Номер протокола	0...3*1	---	0
Point	Номер точки выборки		Номер точки выборки	Зависит от типа данных.		
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Возможные значения для модулей ЦПУ NX102, NX1P2, NJ301 и NJ101: *0 и 1*.

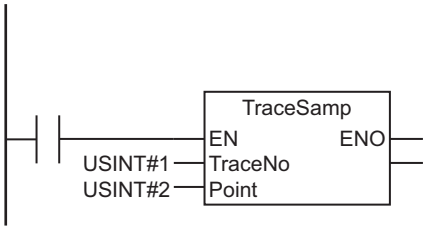
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TraceNo						OK														
Point						OK														
Out	OK																			

Функция

Команда TraceSamp выполняет отбор данных для протокола данных.
Параметры отбора данных задаются в Sysmac Studio. При выполнении этой команды считываются текущие значения всех переменных, для которых настроен отбор данных. Считанные значения сохраняются в память протокола данных вместе с указанным номером протокола (*TraceNo*) и указанным номером точки выборки (*Point*).
Эта команда выполняется только во время выполнения протоколирования данных и только тогда, когда в настройках периода отбора данных в Sysmac Studio выбрано **Use sampling instruction (Использовать команду отбора данных)**.

Пример программы представлен на рисунке ниже. Текущие значения всех выбранных для отбора данных переменных вместе с номером протокола 1 и номером точки выборки 2 сохраняются в память протокола данных.

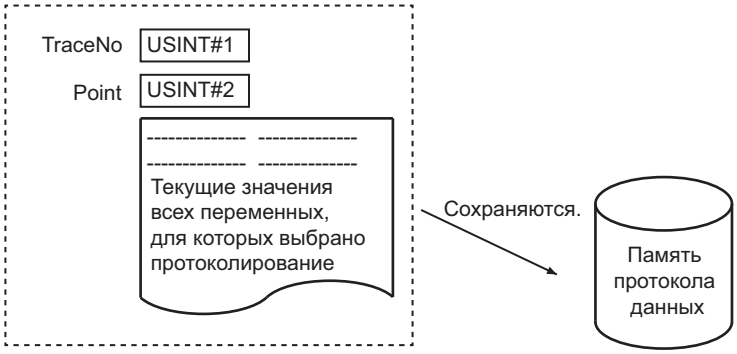
LD



ST

```
TraceSamp(USINT#1, USINT#2);
```

Текущие значения всех переменных, для которых выбрано протоколирование, считываются и сохраняются в память протокола данных с указанием номера протокола (***TraceNo***) и номера точки выборки (***Point***).



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_TraceSta[0..3]*1	Информация о протоколе	_sTRACE_STA[]	Информация о протоколе*2

*1. Модуль ЦПУ NX102, NX1P2, NJ301 или NJ101: имя переменной: _PLC_TraceSta[0..1].

*2. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Дополнительная информация

- Сведения о протоколировании данных см. в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
- Функция протоколирования используется для получения выборки значений указанных переменных. Значения отбираются при наступлении определенных условий. Условия задаются в Sysmac Studio.
- Эту команду можно использовать многократно в разных местах пользовательской программы. Можно составить программу так, чтобы отбор значений производился при выполнении определенных условий.
- Параметр *Point* можно задавать так, чтобы было понятно, с помощью какого именно экземпляра команды TraceSamp были отобраны те или иные данные, отображаемые в окне протокола данных (Data Trace Window) в Sysmac Studio. Если параметр *Point* опущен, по умолчанию используется значение 0.

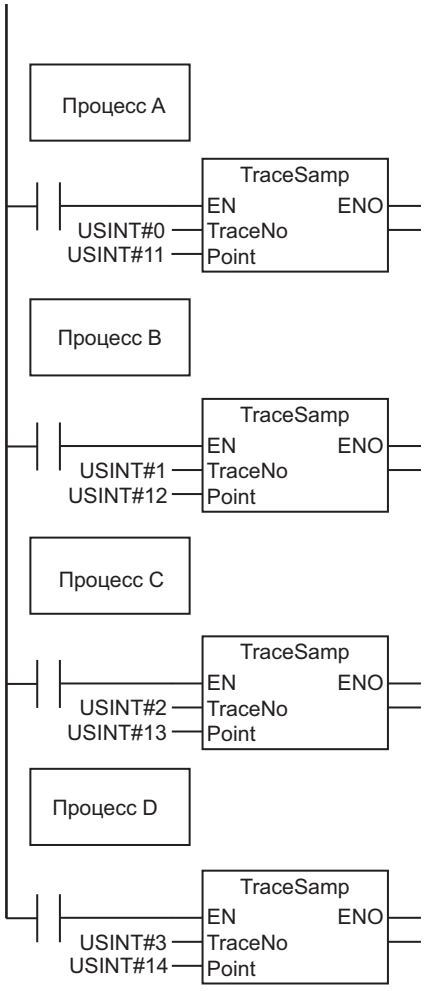
Меры предосторожности для обеспечения надлежащей эксплуатации

- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях ничего не происходит и команда завершается нормально.
  a) Протоколирование данных остановлено.
  b) В параметрах протоколирования для периода отбора значений выбрано не **Use sampling instruction (Использовать команду отбора данных)**.
  c) Значение *TraceNo* не является номером протокола, который задан в Sysmac Studio.
- В указанном ниже случае происходит ошибка. Выход *ENO* перейдет в состояние ЛОЖЬ.
  a) Значение *TraceNo* находится за пределами допустимого диапазона.

Пример программы

В рассматриваемом примере отбор значений производится в конце каждого процесса (A...D). Сохраняются значения переменных в каждой точке.

Программа на языке LD



Программа на языке ST

```
Процесс A
TraceSamp(USINT#0, USINT#11);
Процесс B
TraceSamp(USINT#1, USINT#12);
Процесс C
TraceSamp(USINT#2, USINT#13);
Процесс D
TraceSamp(USINT#3, USINT#14);
```

TraceTrig

Команда TraceTrig активирует событие запуска для протоколирования данных.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
TraceTrig	Запуск протокола данных	FUN	(@)TraceTrig EN ENO TraceNo Out	TraceTrig(TraceNo);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
TraceNo	Номер протокола	Вход	Номер протокола	0...3*1	---	0
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Возможные значения для модулей ЦПУ NX102, NX1P2, NJ301 и NJ101: *0 и 1*.

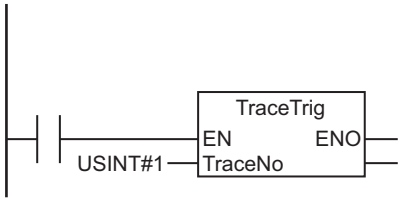
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TraceNo						ОК														
Out	ОК																			

Функция

Команда TraceTrig активирует событие запуска для протоколирования данных.
Для ее работы не имеет значения, были ли выполнены условия запуска, заданные в Sysmac Studio.
При выполнении команды начинается отбор данных для протокола с номером *TraceNo* при условии, что в данный момент производится протоколирование данных.

Пример программы представлен на рисунке ниже. В данном случае событие запуска протокола данных активируется для протокола с номером 1.

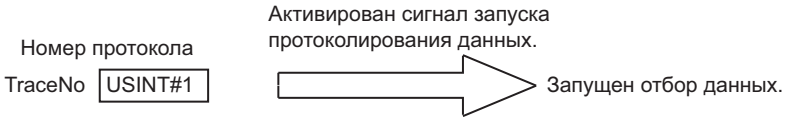
LD



ST

```
TraceTrig(USINT#1);
```

Активируется сигнал запуска протоколирования данных для протокола с номером ***TraceNo***.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_TraceSta[0..3]*1	Информация о протоколе	_sTRACE_STA[]	Информация о протоколе*2

*1. Модуль ЦПУ NX102, NX1P2, NJ301 или NJ101: имя переменной: _PLC_TraceSta[0..1].

*2. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Дополнительная информация

- Дополнительные сведения о протоколировании данных см. в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
- Эту команду можно использовать многократно в разных местах пользовательской программы. Можно составить программу так, чтобы событие запуска активировалось в зависимости от тех или иных условий.
- С помощью этой команды можно программировать условия запуска, которые невозможно настроить с помощью обычных параметров условий запуска. Например, можно предусмотреть активацию события запуска в зависимости от результатов сравнения двух переменных.

Меры предосторожности для обеспечения надлежащей эксплуатации

- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях ничего не происходит и команда завершается нормально.
  a) Протоколирование данных остановлено.
  b) Условие запуска уже выполнено.
  c) Значение *TraceNo* не является номером протокола, который задан в Sysmac Studio.
  d) Для номера протокола, который указан с помощью *TraceNo*, в качестве типа протоколирования выбрано **Continuous trace (Непрерывное протоколирование)**.
- В указанном ниже случае произойдет ошибка. Выход *ENO* перейдет в состояние ЛОЖЬ.
  a) Значение *TraceNo* находится за пределами допустимого диапазона.

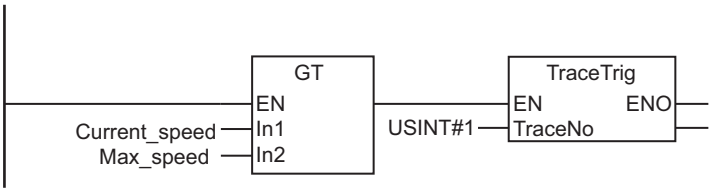
Пример программы

В данном примере событие запуска протокола данных активируется с целью сохранения значений переменных, когда текущая скорость оказывается выше максимально допустимого значения.

Когда значение *Current_speed* превышает значение *Max_speed*, выполняется команда TraceTrig.

Программа на языке LD

Переменная	Тип данных	Начальное значение	Комментарий
Current_speed	INT	0	Текущая скорость
Max_speed	INT	20	Максимальная скорость



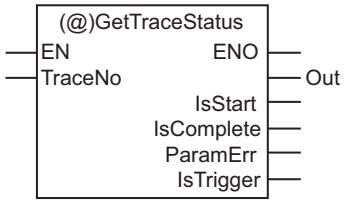
Программа на языке ST

Переменная	Тип данных	Начальное значение	Комментарий
Current_speed	INT	0	Текущая скорость
Max_speed	INT	20	Максимальная скорость

```
IF (Current_speed > Max_speed) THEN
  TraceTrig(USINT#1);
END_IF;
```

GetTraceStatus

Команда GetTraceStatus считывает состояние выполнения протокола данных.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetTraceStatus	Чтение состояния протокола данных	FUN		GetTraceStatus(TraceNo, IsStart, IsComplete, ParamErr, IsTrigger);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
TraceNo	Номер протокола	Вход	Номер протокола	0...3*1	---	0
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---
IsStart	Флаг выполнения		ИСТИНА: производится протоколирование данных. ЛОЖЬ: не производится протоколирование данных.	Зависит от типа данных.		
IsComplete	Флаг завершения		ИСТИНА: протоколирование данных было завершено. ЛОЖЬ: протоколирование данных выполняется в данный момент или не выполнялось.			
ParamErr	Флаг ошибки параметра		ИСТИНА: ошибка настройки протоколирования данных. ЛОЖЬ: нет ошибок настройки протоколирования данных.			
IsTrigger	Флаг события запуска		ИСТИНА: условие запуска протоколирования данных соблюдено. ЛОЖЬ: условие запуска протоколирования данных не соблюдено.			

*1. Возможные значения для модулей ЦПУ NX102, NX1P2, NJ301 и NJ101: *0 и 1*.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TraceNo						OK														
Out	OK																			
IsStart	OK																			
IsComplete	OK																			
ParamErr	OK																			
IsTrigger	OK																			

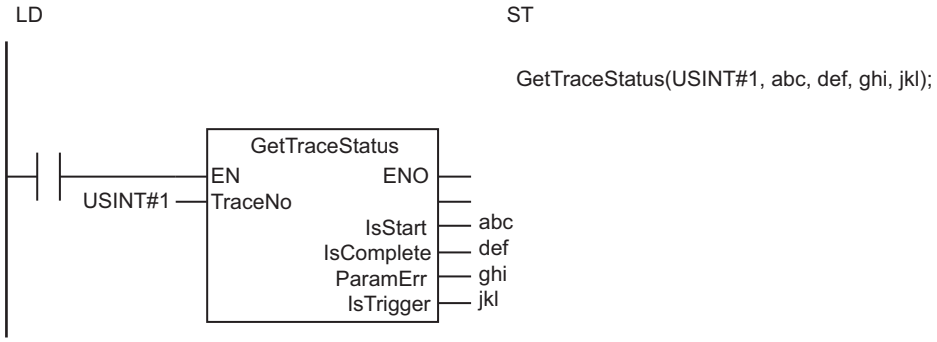
Функция

Команда GetTraceStatus считывает состояние выполнения для протокола данных, номер которого указан параметром *TraceNo*.
Считанное состояние выдается в параметры *IsStart* (Флаг выполнения), *IsComplete* (Флаг завершения), *ParamErr* (Флаг ошибки параметра) и *IsTrigger* (Флаг события запуска).

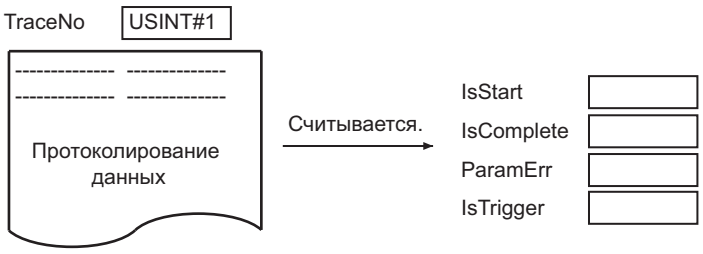
Значение флага *ParamErr* меняется на ИСТИНА при обнаружении в параметрах протокола данных любой из указанных ниже ошибок.

- В параметрах условий запуска или отбора данных указана несуществующая переменная.
- В настройках периода отбора данных выбран **период выполнения указанной задачи**, но этой задачи не существует.

Пример программы представлен на рисунке ниже. Команда GetTraceStatus считывает состояние выполнения для протокола данных с номером протокола 1.



Команда GetTraceStatus считывает состояние выполнения протоколирования данных для указанного номера протокола ***TraceNo***.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_TraceSta[0..3]*1	Информация о протоколе	_sTRACE_STA[]	Информация о протоколе*2

*1. Модуль ЦПУ NX102, NX1P2, NJ301 или NJ101: имя переменной: _PLC_TraceSta[0..1].
*2. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Дополнительная информация

Дополнительные сведения о протоколировании данных см. в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

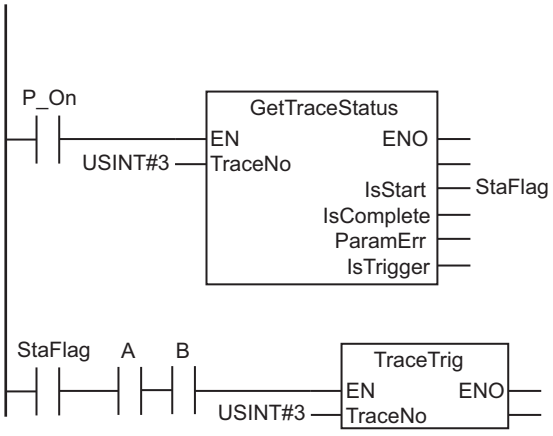
- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- Данная команда считывает содержимое системной переменной _PLC_TraceSta[]. К этой переменной невозможно обратиться напрямую. Для чтения содержимого этой переменной всегда следует использовать данную команду.
- В указанном ниже случае произойдет ошибка. Выход *ENO* перейдет в состояние ЛОЖЬ.
  a) Значение *TraceNo* находится за пределами допустимого диапазона.

Пример программы

В данном примере команда GetTraceStatus считывает состояние выполнения для протокола данных с номером протокола 3. Если в данный момент производится протоколирование данных, выполняется команда TraceTrig для активации протоколирования данных.

Программа на языке LD

Переменная	Тип данных	Начальное значение	Комментарий
StaFlag	BOOL	ЛОЖЬ	Состояние выполнения протоколирования
A	BOOL	ЛОЖЬ	
B	BOOL	ЛОЖЬ	



Программа на языке ST

Переменная	Тип данных	Начальное значение	Комментарий
StaFlag	BOOL	ЛОЖЬ	Состояние выполнения протоколирования
A	BOOL	ЛОЖЬ	
B	BOOL	ЛОЖЬ	

```
GetTraceStatus(TraceNo:=USINT#3, IsStart=>StaFlag);
IF ( (StaFlag=TRUE) AND (A=TRUE) AND (B=TRUE) ) THEN
  TraceTrig(TraceNo:=USINT#3);
END_IF;
```

SetAlarm

Команда SetAlarm создает ошибку пользователя.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SetAlarm	Создание ошибки пользователя	FUN	(@)SetAlarm EN ENO Code Out Info1 Info2	SetAlarm(Code, Info1, Info2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Code	Код события	Вход	Код события генерируемой ошибки пользователя	1…40 000	---	1
Info1	Прилагаемая информация 1		Значения, записываемые в журнал событий при активации ошибки пользователя	Зависит от типа данных.		*1
Info2	Прилагаемая информация 2					
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Code							OK													
Info1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
Info2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
Out	OK																			

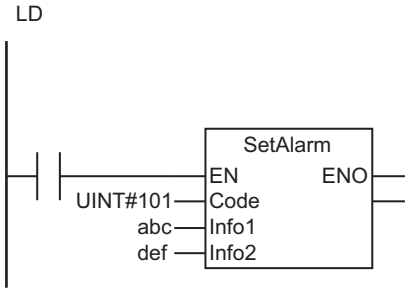
Функция

Команда SetAlarm генерирует определенную пользователем ошибку, которая соответствует коду события *Code*. Коды событий определяются в таблице настройки событий в Sysmac Studio.
В область журнала событий пользователя, которая соответствует уровню кода события, сохраняются следующие данные: время возникновения, имя события, группа событий, код события *Code*, уровень события, дополнительная информация *Info1*, дополнительная информация *Info2* и подробная информация. Значение времени возникновения получается автоматически. Записываются имя события, группа событий и подробная информация, заданные в Sysmac Studio.

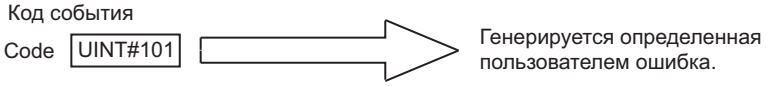
Записывается уровень события, соответствующий коду события. Уровни событий приведены в таблице ниже. Чем меньше код события, тем выше уровень события.

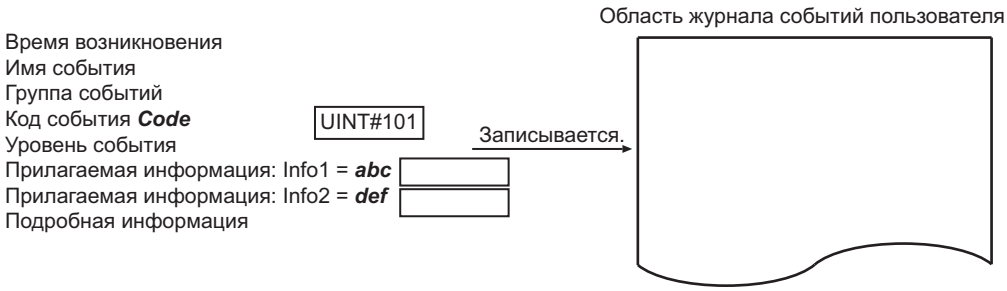
Код события	Классификация: (уровень ошибки пользователя)
1…5000	1
5001...10000	2
10001...15000	3
15001...20000	4
20001...25000	5
25001...30000	6
30001...35000	7
35001...40000	8

Пример программы представлен на рисунке ниже. Создается ошибка пользователя с кодом события 101. В качестве прилагаемой информации сохраняются значения переменных *abc* и *def*.



Генерируется пользовательская ошибка с кодом события ***Code***.
Кроме того, в область журнала событий пользователя сохраняется время возникновения, имя события, группа событий, код события ***Code***, уровень события, дополнительная информация ***Info1***, дополнительная информация ***Info2*** и подробная информация.





Связанные системные переменные

Имя	Значение	Тип данных	Описание
_AlarmFlag	Состояния ошибок, определяемых пользователем	WORD	Эти флаги указывают на обнаружение ошибок, определенных пользователем. Биты 0...7 указывают состояние ошибки пользователя уровня 1...8 соответственно. *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Дополнительная информация

Для *Info1* и *Info2* можно указать глобальные переменные или локальные переменные.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Для каждого из восьми уровней события может быть сгенерировано до 32 ошибок пользователя (всего до 256 ошибок пользователя).
- Если ошибка пользователя с таким кодом события уже существует, новая ошибка в журнале событий не регистрируется.
- Обязательно используйте переменные для входных параметров, передаваемых в *Info1* и *Info2*. Если вместо переменных будут указаны константы, произойдет ошибка сборки.
- Даже если в параметр *Code* будет передан код события, не зарегистрированный в Sysmac Studio, ошибки не произойдет. Если код события не зарегистрирован, то группа событий и подробная информация в журнал событий пользователя не записываются. В качестве имени события записывается значение *Code*.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* перейдет в состояние ЛОЖЬ.
  a) Значение *Code* находится за пределами допустимого диапазона.
  b) Была предпринята попытка сгенерировать больше ошибок пользователя, чем это возможно.

Пример программы

В этом примере переменная *A* каждые пять секунд переключается между состояниями ИСТИНА и ЛОЖЬ.
Значение переменной *A* контролируется. Если оно не изменяется дольше пяти секунд, генерируется пользовательская ошибка с кодом события 102. В качестве прилагаемой информации указываются значения UINT#123 и UINT#456.
Когда значение переменной *F* меняется на ИСТИНА, ошибка пользователя очищается.

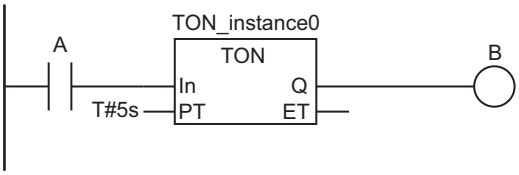
Программа на языке LD

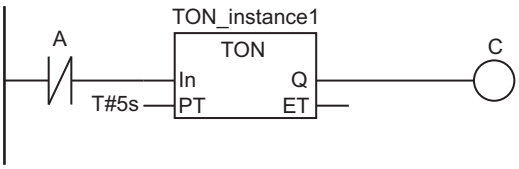
Внутренние переменные	Переменная	Тип данных	Начальное значение
	A	BOOL	ЛОЖЬ
	B	BOOL	ЛОЖЬ
	C	BOOL	ЛОЖЬ

Внутренние переменные	Переменная	Тип данных	Начальное значение
	F	BOOL	ЛОЖЬ
	Abc	UINT	123
	Def	UINT	456
	TON_instance0	TON	
	TON_instance1	TON	

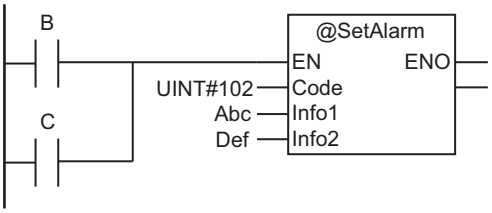
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_AlarmFlag	WORD	☑	Состояния ошибок, определяемых пользователем

Проверка значения переменной ***A***.

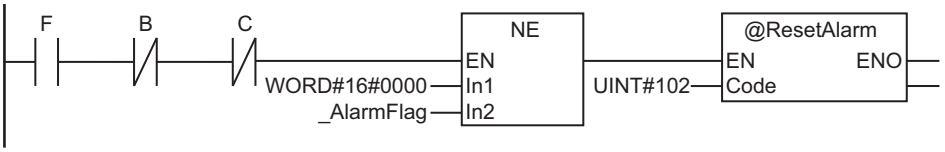




Создание ошибки пользователя.



Сброс ошибки пользователя.



Программа на языке ST

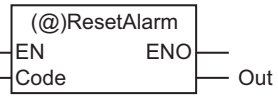
Внутренние переменные	Переменная	Тип данных	Начальное значение
	A	BOOL	ЛОЖЬ
	B	BOOL	ЛОЖЬ
	C	BOOL	ЛОЖЬ
	F	BOOL	ЛОЖЬ
	Abc	UINT	123
	Def	UINT	456
	TON_instance0	TON	
	TON_instance1	TON	

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_AlarmFlag	WORD	☑	Состояния ошибок, определяемых пользователем

```
// Проверка значения переменной A.
IF (A=TRUE) THEN
  TON_instance0(In:=TRUE, PT:=T#5s, Q=>B);
ELSE
  TON_instance0(In:=FALSE, Q=>B);
END_IF;

IF (A=FALSE) THEN
  TON_instance1(In:=TRUE, PT:=T#5s, Q=>C);
ELSE
  TON_instance1(In:=FALSE, Q=>C);
END_IF;

// Создание ошибки пользователя
IF (B=TRUE) OR (C=TRUE) THEN
  SetAlarm(
    Code:=UINT#102,
    Info1 :=Abc,
    info2 :=Def);
END_IF;

// Сброс ошибки пользователя.
IF (F=TRUE) & (B=FALSE) & (C=FALSE) & (_AlarmFlag<>WORD#16#0000) THEN
  ResetAlarm(Code:=UINT#102);
END_IF;
```

ResetAlarm

Команда ResetAlarm сбрасывает ошибку пользователя.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ResetAlarm	Сброс ошибки пользователя	FUN		ResetAlarm(Code);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Code	код события	Вход	Код события сбрасываемой ошибки пользователя 16#0: сбросить все ошибки приложений.	0...40000	---	0
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

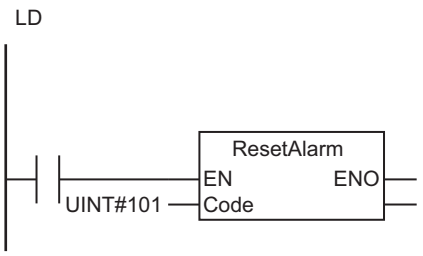
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Code							OK													
Out	OK																			

Функция

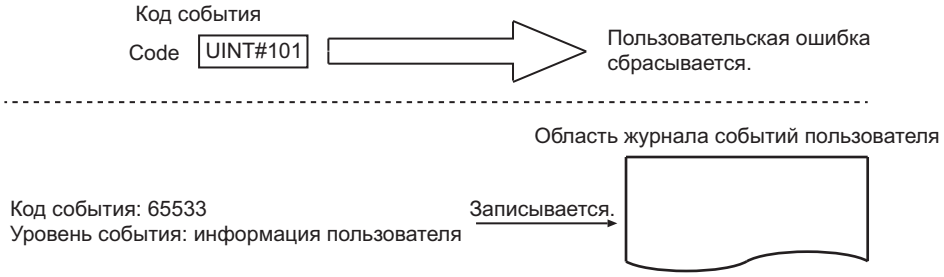
Команда ResetAlarm сбрасывает определенную пользователем ошибку, указанную параметром *Code*.
При этом в область журнала событий пользователя записывается событие, указывающее, что была сброшена конкретная пользовательская ошибка. Этому событию соответствуют код события 65533 и уровень "Информация пользователя".

Если *Code* = 0, то сбрасываются все текущие пользовательские ошибки.
При этом в область журнала событий пользователя записывается событие, указывающее, что были сброшены все пользовательские ошибки. Этому событию соответствуют код события 65534 и уровень "Информация пользователя".

Пример программы представлен на рисунке ниже. Сбрасывается ошибка пользователя с кодом события 101.



Команда ResetAlarm сбрасывает пользовательскую ошибку, указанную кодом события ***Code***.
Также в область журнала событий пользователя записывается событие, показывающее, что была сброшена конкретная пользовательская ошибка.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_AlarmFlag	Состояния ошибок, определяемых пользователем	WORD	Эти флаги указывают на обнаружение ошибок, определенных пользователем. Биты 0...7 указывают состояние ошибки пользователя уровня 1...8 соответственно. *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

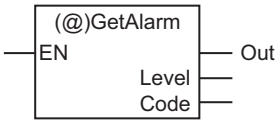
- Если ошибки пользователя, указанной значением *Code*, еще не произошло, к ошибке команды это не приводит.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* перейдет в состояние ЛОЖЬ.
  a) Значение *Code* находится за пределами допустимого диапазона.

Пример программы

См. *Пример программы* на стр. 2-938 для команды SetAlarm.

GetAlarm

Команда GetAlarm возвращает наивысший уровень события (среди уровней ошибки пользователя 1...8) и код события наивысшего уровня среди текущих ошибок пользователя.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetAlarm	Получить состояние ошибки пользователя	FUN		Out:=GetAlarm(Level, Code);

Переменные

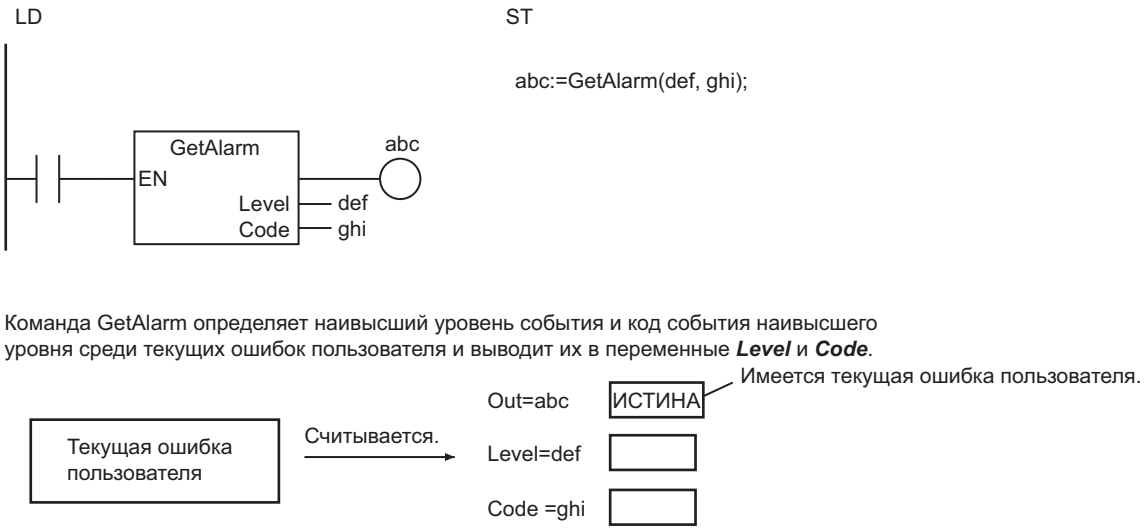
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Флаг ошибки	Выход	ИСТИНА: имеется ошибка пользователя. ЛОЖЬ: ошибки пользователя отсутствуют.	Зависит от типа данных.	---	---
Level	Наивысший уровень события		Наиболее высокий уровень события среди всех текущих ошибок пользователя 0: ошибки пользователя отсутствуют. 1...8: уровень события	0...8		
Code	Код события наивысшего уровня		Код события наиболее высокого уровня среди всех текущих ошибок пользователя 0: ошибки пользователя отсутствуют. 1...40 000: код события	0...40000		

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	ОК																			
Level							ОК													
Code							ОК													

Функция

Команда GetAlarm определяет наивысший уровень события и код события наивысшего уровня среди текущих ошибок пользователя и выводит их в переменные *Level* и *Code*.
Если в данный момент ошибки пользователя отсутствуют, переменная *Out* (Флаг ошибки) будет содержать значение ЛОЖЬ.
Если имеется несколько ошибок пользователя с данным наиболее высоким уровнем события, в переменную *Code* записывается код события для ошибки пользователя, которая возникла первой.

Пример программы представлен на рисунке ниже.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_AlarmFlag	Состояния ошибок, определяемых пользователем	WORD	Эти флаги указывают на обнаружение ошибок, определенных пользователем. Биты 0...7 указывают состояние ошибки пользователя уровня 1...8 соответственно. *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

ResetPLCError

Команда ResetPLCError сбрасывает ошибки в функциональном модуле «PLC».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ResetPLCError	Сброс ошибки контроллера в PLC	FB	ResetPLCError_instance ResetPLCError Execute Done Busy Error ErrorID	ResetPLCError(Execute, Done, Busy, Error, ErrorID);

Переменные

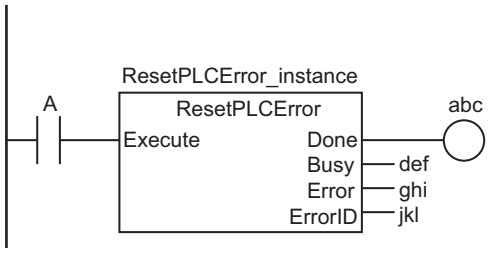
Используются только общие переменные.

Функция

Команда ResetPLCError сбрасывает ошибки в функциональном модуле «PLC».

Пример программы представлен на рисунке ниже.

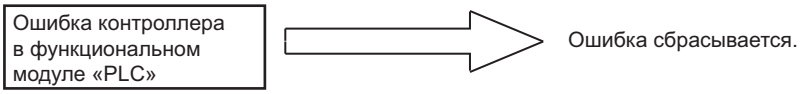
LD



ST

```
ResetPLCError_instance(A, abc, def, ghi, jkl);
```

Команда ResetPLCError сбрасывает ошибки в функциональном модуле «PLC».



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_ErrSta	Состояние ошибки функционального модуля «PLC»	WORD	Содержит состояние ошибки функционального модуля «PLC». *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

После выполнения этой команды ошибка может быть сброшена не сразу, а с некоторой задержкой. Используйте команду GetPLCError, чтобы убедиться, что ошибки были сброшены.

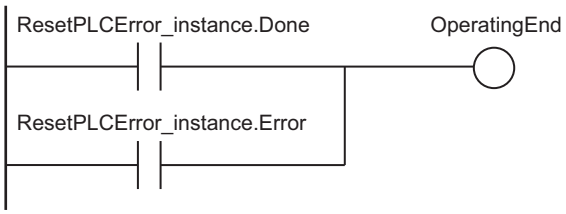
Пример программы

Команда ResetPLCError выполняется, когда значение переменной *Trigger* меняется на ИСТИНА. Если выполнение команды ResetPLCError завершается нормально (т. е. если значение *Done* = ИСТИНА), выполняется обработка нормального завершения. Если же выполнение завершается ошибкой (т. е. если значение *Error* = ИСТИНА), то выполняется обработка завершения с ошибкой.

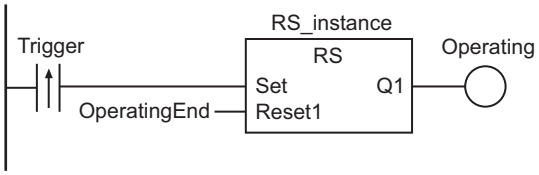
Программа на языке LD

Переменная	Тип данных	Начальное значение	Комментарий
OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена
Trigger	BOOL	ЛОЖЬ	Условие выполнения
Operating	BOOL	ЛОЖЬ	Обработка
RS_instance	RS		
ResetPLCError_instance	ResetPLCError		

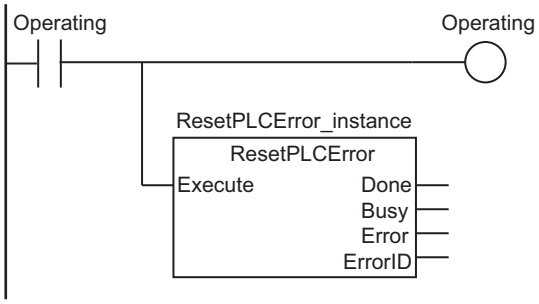
Определение, завершилось ли выполнение ResetPLCError.



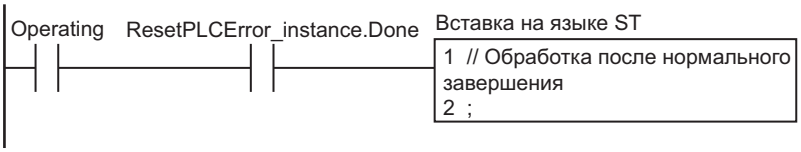
Прием условия выполнения.



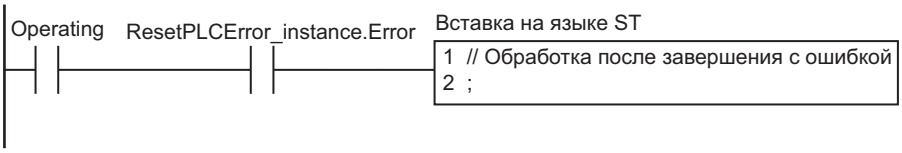
Выполнение команды ResetPLCError.



Обработка после нормального завершения



Обработка после завершения с ошибкой



Программа на языке ST

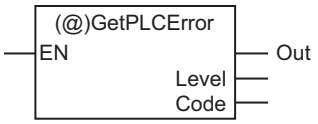
Переменная	Тип данных	Начальное значение	Комментарий
Trigger	BOOL	ЛОЖЬ	Условие выполнения
LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
OperatingStart	BOOL	ЛОЖЬ	Обработка началась
Operating	BOOL	ЛОЖЬ	Обработка
ResetPLCError_instance	ResetPLCError		

```
// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) ) THEN
  OperatingStart:=TRUE;
  Operating:=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация ResetPLCError_instance.
IF (OperatingStart=TRUE) THEN
  ResetPLCError_instance(Execute:=FALSE);
  OperatingStart:=FALSE;
END_IF;

// Выполнение команды ResetPLCError.
IF (Operating=TRUE) THEN
  ResetPLCError_instance(Execute:=TRUE);

  IF (ResetPLCError_instance.Done=TRUE) THEN
    // Обработка после нормального завершения
    Operating:=FALSE;
  END_IF;

  IF (ResetPLCError_instance.Error=TRUE) THEN
    // Обработка после завершения с ошибкой
    Operating:=FALSE;
```

```
    END_IF;
  END_IF;
```

GetPLCError

Команда GetPLCError определяет наиболее высокий уровень события (частично критическая ошибка или некритическая ошибка) и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «PLC».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetPLCError	Получить состояние ошибки контроллера в PLC	FUN		Out:=GetPLCError(Level, Code);

Переменные

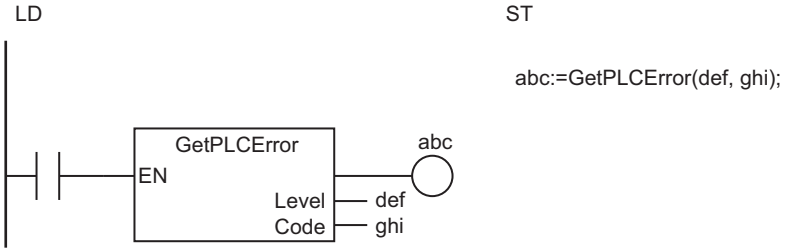
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Флаг ошибки	Выход	ИСТИНА: имеется ошибка контроллера. ЛОЖЬ: ошибки контроллера отсутствуют.	Зависит от типа данных.	---	---
Level	Наивысший уровень события		Наиболее высокий уровень события из всех текущих ошибок контроллера в функциональном модуле «PLC» 0: ошибки контроллера отсутствуют. 2: частично критическая ошибка 3: некритическая ошибка	0, 2 или 3		
Code	Код события наивысшего уровня		Код события наиболее высокого уровня среди всех текущих ошибок контроллера в функциональном модуле «PLC» 16#0000_0000: ошибки контроллера отсутствуют. 16#0007_0000...16#FFFF_FFFF: код события	16#00000000, 16#00070000... 16#FFFFFFFF		

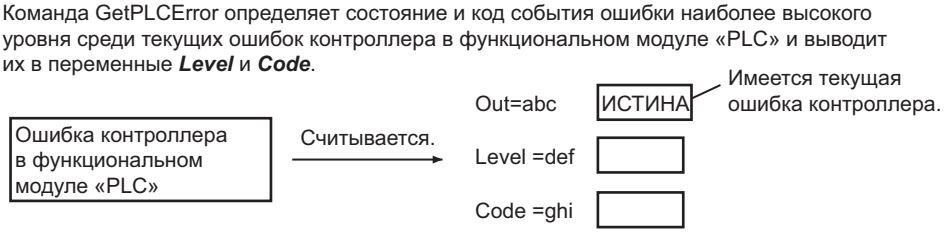
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	ОК																			
Level							ОК													
Code				ОК																

Функция

Команда GetPLCError определяет наиболее высокий уровень события и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «PLC» и выводит их в переменные *Level* и *Code*.
Если в данный момент ошибки контроллера отсутствуют, переменная *Out* (Флаг ошибки) будет содержать значение ЛОЖЬ.
Если имеется несколько ошибок контроллера с данным наиболее высоким уровнем события, в переменную *Code* записывается код события для ошибки контроллера, которая возникла первой.

Пример программы представлен на рисунке ниже.





Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_ErrSta	Состояние ошибки функционального модуля «PLC»	WORD	Содержит состояние ошибки функционального модуля «PLC». *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

ResetCJBError

Команда ResetCJBError сбрасывает ошибки контроллера в шине ввода-вывода.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ResetCJBError	Сброс ошибки контроллера в шине CJ	FB	ResetCJBError_instance ResetCJBError Execute — Done UnitNo — Busy Error ErrorID	ResetCJBError_instance(Execute, UnitNo, Done, Busy, Error, ErrorID);

Меры предосторожности для обеспечения надлежащей эксплуатации

Данную команду невозможно использовать с модулями ЦПУ серии NX.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitNo	Номер модуля	Вход	Номер модуля, для которого нужно сбросить ошибки	_CBU_No00..._CBU_No15, _SIO_No00..._SIO_No95, _UNIT_ALL	---	_UNIT_ALL

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitNo	Сведения о перечислителях перечислимого типа _eUnitNo см. в разделе *Функция* на стр. 2-951.																			

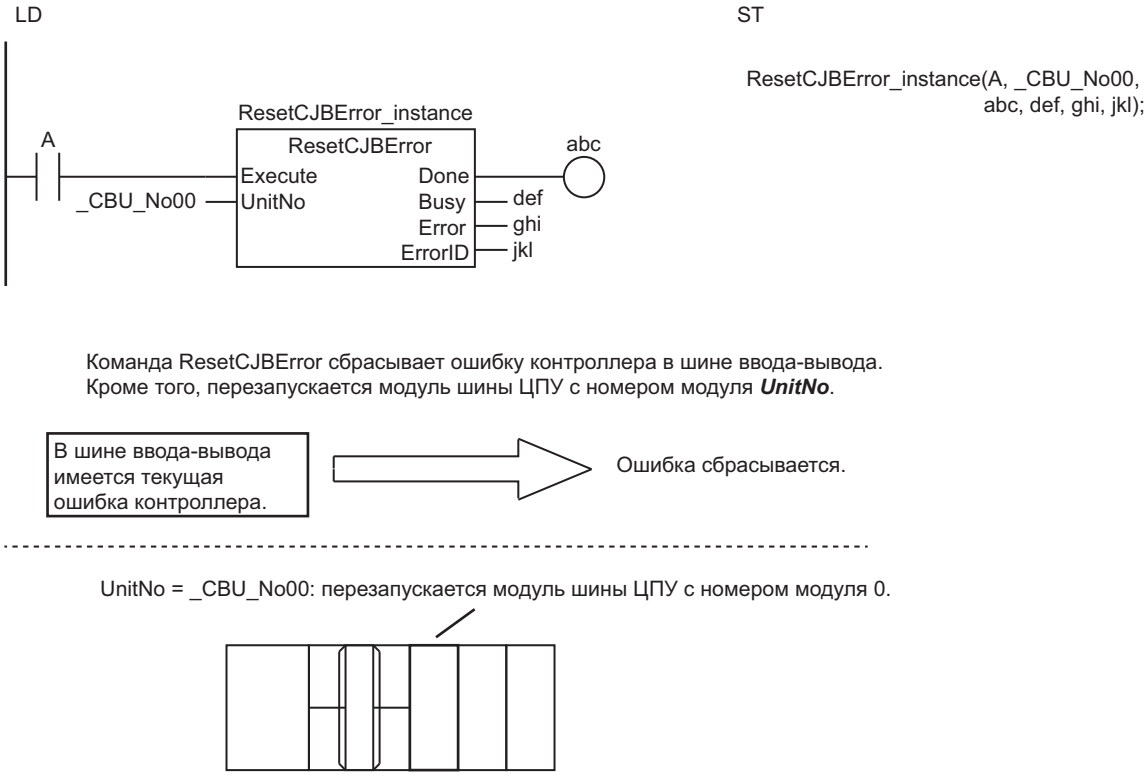
Функция

Команда ResetCJBError сбрасывает ошибку контроллера в шине ввода-вывода.
Если модуль, указанный параметром *UnitNo*, является специальным модулем серии CJ, модуль также перезапускается.

Для параметра *UnitNo* используется перечислимый тип данных _eUnitNo. Значения перечислителей приведены в таблице ниже.

Перечислители	Значение
_CBU_No00..._CBU_No15	Номер модуля шины ЦПУ, 00...15
_SIO_No00..._SIO_No95	Номер модуля специального модуля ввода-вывода, 00...95
_UNIT_ALL	Все модули

Ниже приведен пример для случая, когда *UnitNo* = _CBU_No00. Сбрасывается ошибка контроллера в шине ввода-вывода, и модуль шины ЦПУ с номером модуля 0 перезапускается.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CJB_ErrSta	Состояние ошибки шины ввода-вывода	WORD	Содержит состояние ошибки шины ввода-вывода. *1

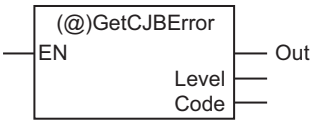
*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- После выполнения этой команды ошибка может быть сброшена не сразу, а с некоторой задержкой. Используйте команду GetCJBError, чтобы убедиться, что ошибки были сброшены.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение *UnitNo* находится за пределами допустимого диапазона.
  b) Модуля с указанным номером *UnitNo* не существует.

GetCJBError

Команда GetCJBError определяет наиболее высокий уровень события и код события наиболее высокого уровня среди текущих ошибок контроллера в шине ввода-вывода модуля ЦПУ серии NJ.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetCJBError	Получить состояние ошибки шины ввода-вывода	FUN		Out:=GetCJBError(Level, Code);

Меры предосторожности для обеспечения надлежащей эксплуатации

Данную команду невозможно использовать с модулями ЦПУ серии NX.

Переменные

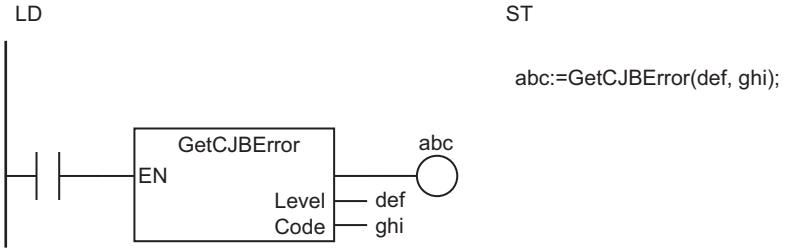
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Флаг ошибки	Выход	ИСТИНА: имеется ошибка контроллера. ЛОЖЬ: ошибки контроллера отсутствуют.	Зависит от типа данных.	---	---
Level	Наивысший уровень события		Наиболее высокий уровень события из всех текущих ошибок контроллера в шине ввода-вывода 0: ошибки контроллера отсутствуют 2: частично критическая ошибка 3: некритическая ошибка	0, 2 или 3		
Code	Код события наивысшего уровня		Код события наиболее высокого уровня среди всех текущих ошибок контроллера в шине ввода-вывода 16#0000_0000: ошибки контроллера отсутствуют. 16#0007_0000...16#FFFF_FFFF: код события	16#00000000, 16#00070000... 16#FFFFFFFF		

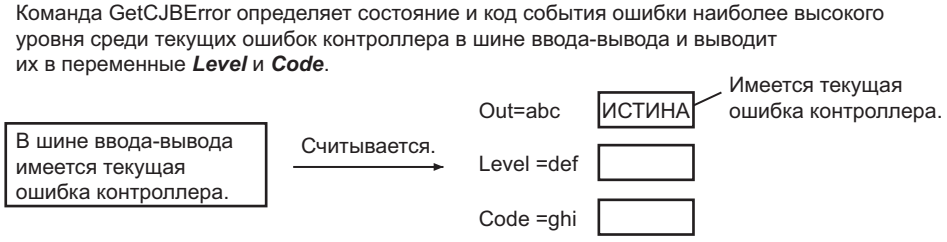
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	ОК																			
Level							ОК													
Code				ОК																

Функция

Команда GetCJBError определяет наиболее высокий уровень события и код события наиболее высокого уровня среди текущих ошибок контроллера в шине ввода-вывода и выводит их в переменные *Level* и *Code*.
Если в данный момент ошибки контроллера отсутствуют, переменная *Out* (Флаг ошибки) будет содержать значение ЛОЖЬ.
Если имеется несколько ошибок контроллера с данным наиболее высоким уровнем события, в переменную *Code* записывается код события для ошибки контроллера, которая возникла первой.

Пример программы представлен на рисунке ниже.





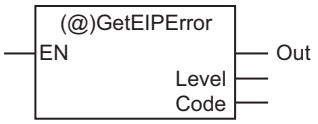
Связанные системные переменные

Имя	Значение	Тип данных	Описание
_CJB_ErrSta	Состояние ошибки шины ввода-вывода	WORD	Содержит состояние ошибки шины ввода-вывода. *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

GetEIPError

Команда GetEIPError определяет наиболее высокий уровень события (частично критическая ошибка или некритическая ошибка) и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «EtherNet/IP».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetEIPError	Получить состояние ошибки в EtherNet/IP	FUN		Out:=GetEIPError(Level, Code);

Переменные

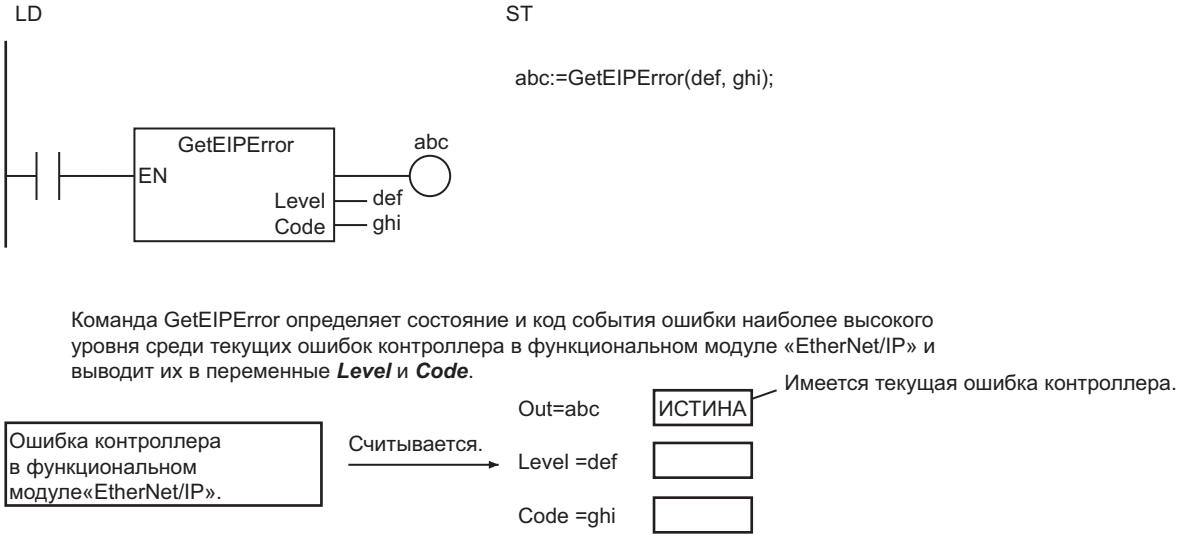
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Флаг ошибки	Выход	ИСТИНА: имеется ошибка контроллера. ЛОЖЬ: ошибки контроллера отсутствуют.	Зависит от типа данных.	---	---
Level	Наивысший уровень события		Наиболее высокий уровень события из всех текущих ошибок контроллера в функциональном модуле «EtherNet/IP» 0: ошибки контроллера отсутствуют. 2: частично критическая ошибка 3: некритическая ошибка	0, 2 или 3		
Code	Код события наивысшего уровня		Код события наиболее высокого уровня среди всех текущих ошибок контроллера в функциональном модуле «EtherNet/IP» 16#0000_0000: ошибки контроллера отсутствуют. 16#0007_0000...16#FFFF_FFFF: код события	16#00000000, 16#00070000... 16#FFFFFFFF		

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	ОК																			
Level							ОК													
Code				ОК																

Функция

Команда GetEIPError определяет наиболее высокий уровень события и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «EtherNet/IP» и выводит их в переменные *Level* и *Code*.
Если в данный момент ошибки контроллера отсутствуют, переменная *Out* (Флаг ошибки) будет содержать значение ЛОЖЬ.
Если имеется несколько ошибок контроллера с данным наиболее высоким уровнем события, в переменную *Code* записывается код события для ошибки контроллера, которая возникла первой.

Пример программы представлен на рисунке ниже.



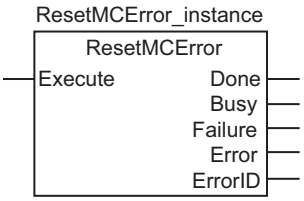
Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_ErrSta	Состояние ошибки функционального модуля «EtherNet/IP»	WORD	Содержит состояние ошибки функционального модуля «EtherNet/IP». *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

ResetMCError

Команда ResetMCError сбрасывает ошибки контроллера в функциональном модуле «Motion Control».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ResetMCError	Сброс ошибки в Motion Control	FB		ResetMCError_instance(Execute, Done, Busy, Failure Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Failure	Завершение со сбоем	Выход	ИСТИНА: ошибки не были сброшены. ЛОЖЬ: ошибки были нормально сброшены.	Зависит от типа данных.	---	---

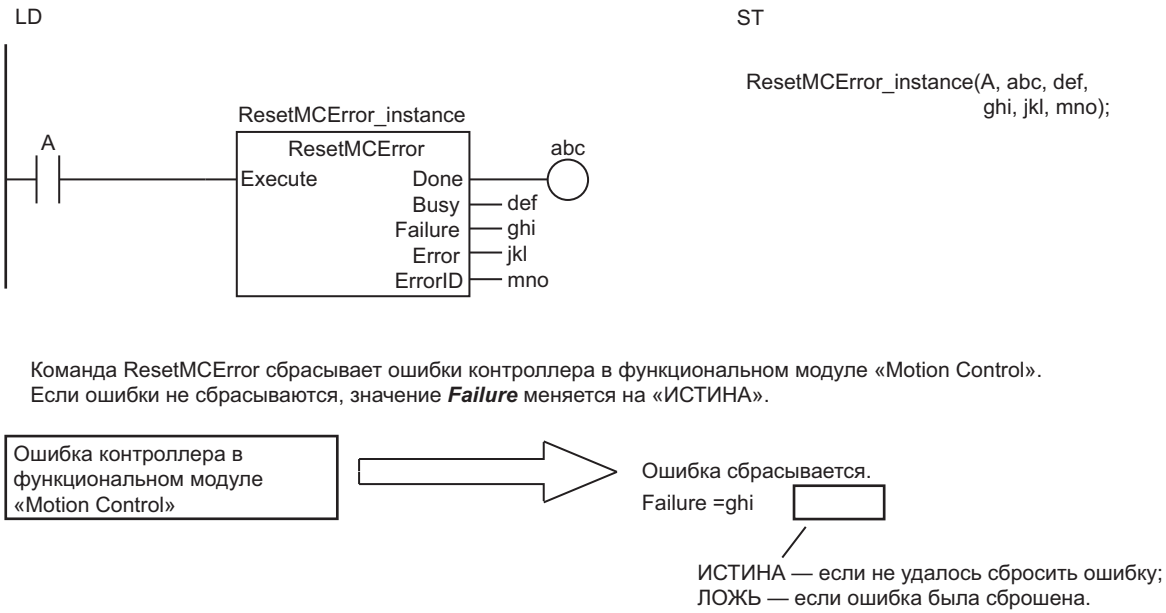
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Failure	ОК																			

Функция

Команда ResetMCError сбрасывает ошибку контроллера в функциональном модуле «Motion Control».
Если ошибки не сбрасываются, значение переменной *Failure* меняется на ИСТИНА.

Действие команды ResetMCError не зависит от того, в какой задаче находится программа, в которой она выполняется. В любом случае эта команда сбрасывает ошибки для всех осей и всех групп осей.

Пример программы представлен на рисунке ниже.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_MC_ErrSta	Состояние ошибки управления движением	WORD	Содержит состояние ошибки функционального модуля «Motion Control». *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- После выполнения этой команды ошибка может быть сброшена не сразу, а с некоторой задержкой. Используйте команду GetMCError, чтобы убедиться, что ошибки были сброшены.
- При попытке выполнить эту команду во время пробного запуска функции MC переменная *Busy* остается в состоянии ИСТИНА и команда не выполняется.
- Если эту команду нужно выполнить для сервопривода OMRON серии G5, используйте эксклюзивное управление для команд, чтобы исключить одновременное выполнение команды ResetECError с этой командой. Если команды ResetMCError и ResetECError будут выполнены одновременно, сервопривод серии G5 больше не будет принимать сообщения SDO.

Информация о версии

- При использовании модуля ЦПУ с версией модуля от 1.02 до 1.09 можно создать только 100 экземпляров этой команды.
- Если в контроллер с модулем ЦПУ с версией модуля от 1.02 до 1.09 будет передана программа пользователя, содержащая более 100 экземпляров этой команды, произойдет ошибка контроллера. Ошибка контроллера зависит от используемого способа передачи программы пользователя, что отражено в таблице ниже.

Способ передачи программы пользователя	Код события для ошибки контроллера	Уровень ошибки контроллера
Передача программы с помощью операции синхронизации	10250000 hex	Критическая ошибка
	571D0000 hex	Наблюдение
Передача программы пользователя путем онлайн-редактирования	571D0000 hex	Наблюдение

- Если программа пользователя, содержащая более 100 экземпляров этой команды, будет передана в контроллер с модулем ЦПУ с версией модуля 1.01 или более ранней версии, указанной выше ошибки контроллера не произойдет. Но если будет создано слишком много экземпляров этой команды, программа пользователя станет слишком большой и возникнет ошибка контроллера критического уровня.

Пример программы

Рассматриваемая в данном примере программа обнаруживает ошибки контроллера в функциональном модуле «EtherCAT Master» и в функциональном модуле «Motion Control». Обнаруживаемые ошибки сбрасываются.

Обработка выполняется в следующем порядке:

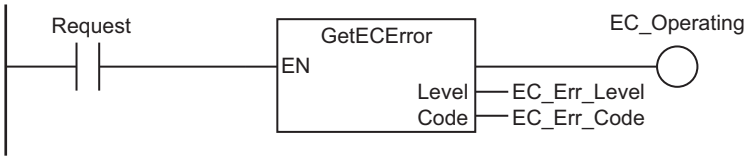
1. Выполняется команда GetECError для обнаружения любых ошибок контроллера в функциональном модуле «EtherCAT Master».
2. Если обнаружены ошибки, они сбрасываются с помощью команды ResetECError.
3. Выполняется команда GetMCError для обнаружения любых ошибок контроллера в функциональном модуле «Motion Control».
4. Если обнаружены ошибки, они сбрасываются с помощью команды ResetMCError.

Программа на языке LD

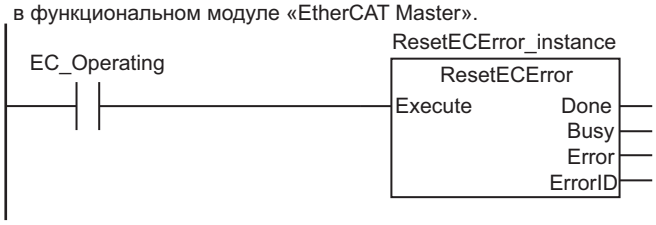
Переменная	Тип данных	Начальное значение	Комментарий
Request	BOOL	ЛОЖЬ	Запрос на обнаружение и сброс ошибок
EC_Err_Level	UINT	0	Функциональный модуль «EtherCAT Master» Наивысший уровень события
EC_Err_Code	DWORD	DWORD#16#0	Функциональный модуль «EtherCAT Master» Код события наивысшего уровня
EC_Operating	BOOL	ЛОЖЬ	Сброс ошибки в функциональном модуле «EtherCAT Master».

Переменная	Тип данных	Начальное значение	Комментарий
MC_Err_Level	UINT	0	Функциональный модуль «Motion Control» Наивысший уровень события
MC_Err_Code	DWORD	DWORD#16#0	Функциональный модуль «Motion Control» Код события наивысшего уровня
MC_Operating	BOOL	ЛОЖЬ	Сброс ошибки в функциональном модуле «Motion Control».
Normal_End	BOOL	ЛОЖЬ	Нормальное завершение
ResetECError_instance	ResetECError		
ResetMCError_instance	ResetMCError		

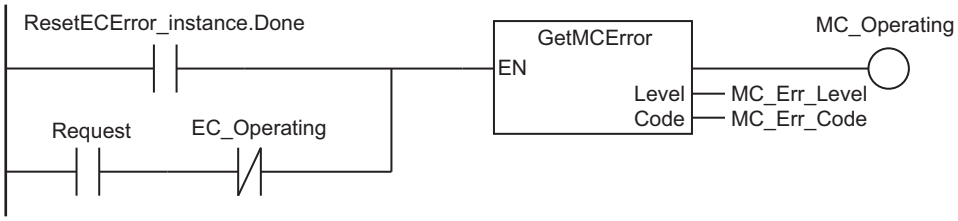
Выполнение команды GetECError.



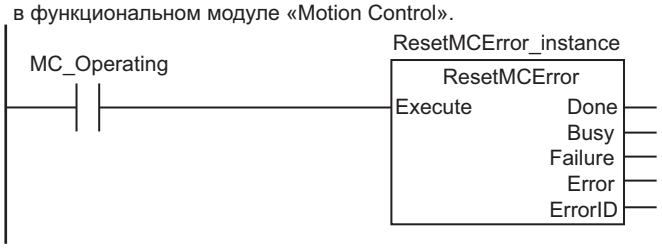
Выполнение команды ResetECError при возникновении ошибки в функциональном модуле «EtherCAT Master».

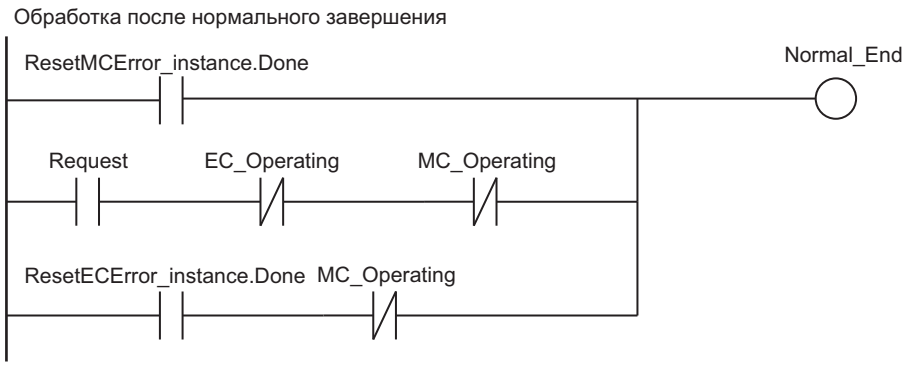


Выполнение команды GetMCError после сброса ошибки в функциональном модуле «EtherCAT Master» или при отсутствии ошибки.



Выполнение команды ResetMCError при возникновении ошибки в функциональном модуле «Motion Control».





Программа на языке ST

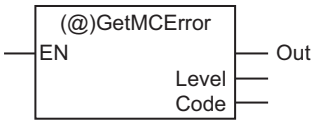
Переменная	Тип данных	Начальное значение	Комментарий
Request	BOOL	ЛОЖЬ	Запрос на обнаружение и сброс ошибок
EC_Error	BOOL	ЛОЖЬ	Ошибка в функциональном модуле «EtherCAT Master»
EC_Err_Level	UINT	0	Наивысший уровень события в функциональном модуле «EtherCAT Master»
EC_Err_Code	DWORD	DWORD#16#0	Код события наивысшего уровня в функциональном модуле «EtherCAT Master»
EC_Stage	INT	0	Сброс ошибки в функциональном модуле «EtherCAT Master»
MC_Error	BOOL	ЛОЖЬ	Ошибка в функциональном модуле «Motion Control»
MC_Err_Level	UINT	0	Наивысший уровень события в функциональном модуле «Motion Control»
MC_Err_Code	DWORD	DWORD#16#0	Код события наивысшего уровня в функциональном модуле «Motion Control»
MC_Stage	INT	0	Сброс ошибки в функциональном модуле «Motion Control»
ResetECError_instance	ResetECError		
ResetMCError_instance	ResetMCError		

```
// Определение запросов на сброс ошибок.
IF (Request=TRUE) THEN

// Обнаружение ошибок контроллера в функц. модуле «EtherCAT Master».
  EC_Error:=GetECError(EC_Err_Level, EC_Err_Code);

// Обнаружение ошибок контроллера в функц. модуле «Motion Control».
  MC_Error:=GetMCError(MC_Err_Level, MC_Err_Code);

  IF (EC_Error=TRUE) THEN // Ошибка контроллера в функц. модуле «EtherCAT Master».
    CASE EC_Stage OF
    0 : // Инициализация
      ResetECError_instance(Execute:=FALSE);
      EC_Stage:=INT#1;
    1 : // Сброс ошибки контроллера в функциональном модуле «EtherCAT Master».
      ResetECError_instance(Execute:=TRUE);
```

```
      IF (ResetECError_instance.Done=TRUE) THEN
        EC_Stage:=INT#99; // Нормальное завершение
      END_IF;
      IF (ResetECError_instance.Error=TRUE) THEN
        EC_Stage:=INT#98; // Завершение с ошибкой
      END_IF;
    99 : // Обработка после нормального завершения
      EC_Stage:=INT#0;
    98 : // Обработка после завершения с ошибкой.
      EC_Stage:=INT#0;
    END_CASE;
  END_IF;

  IF (MC_Error=TRUE) THEN // Ошибка контроллера в функциональном модуле «Motion Con
trol».
    CASE MC_Stage OF
    0 : // Инициализация
      ResetMCError_instance(Execute:=FALSE);
      MC_Stage:=INT#1;
    1 : // Сброс ошибки контроллера в функциональном модуле «Motion Control».
      IF (EC_Error=FALSE) THEN
        ResetMCError_instance(Execute:=TRUE); // Возобновление работы для всех ведо
мых устройств.
        IF (ResetMCError_instance.Done=TRUE) THEN
          MC_Stage:=INT#99; // Нормальное завершение
        END_IF;
        IF ( (ResetMCError_instance.Error=TRUE) OR (ResetMCError_instance.Failure=T
RUE) ) THEN
          MC_Stage:=INT#98; // Завершение с ошибкой
        END_IF;
      END_IF;
    99 : // Обработка после нормального завершения
      MC_Stage:=INT#0;
    98 : // Обработка после завершения с ошибкой.
      MC_Stage:=INT#0;
    END_CASE;
  END_IF;
END_IF;
```

GetMCError

Команда GetMCError определяет наиболее высокий уровень события (частично критическая ошибка или некритическая ошибка) и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «Motion Control».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetMCError	Получить состояние ошибки в Motion Control	FUN		Out:=GetMCError(Level, Code);

Переменные

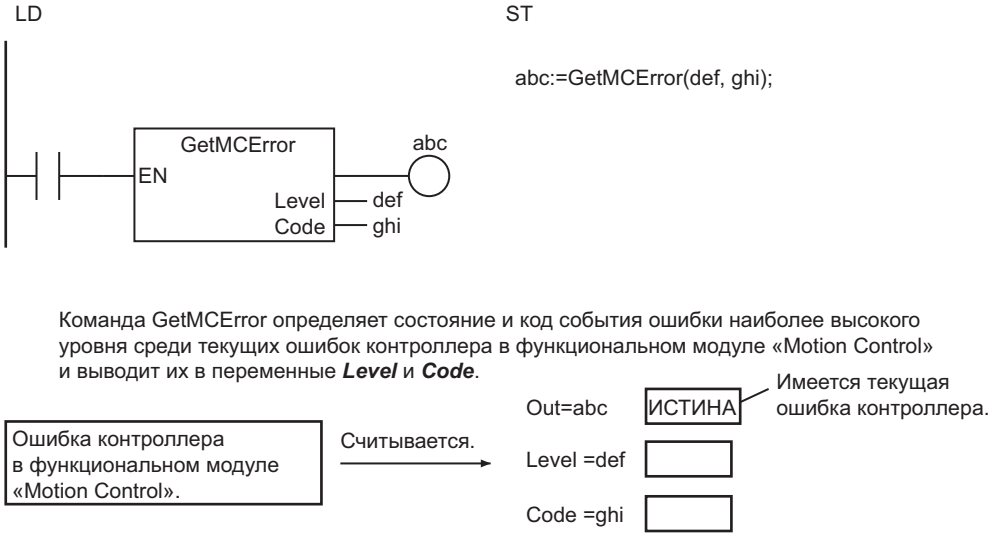
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Флаг ошибки	Выход	ИСТИНА: имеется ошибка контроллера. ЛОЖЬ: ошибки контроллера отсутствуют.	Зависит от типа данных.	---	---
Level	Наивысший уровень события		Наиболее высокий уровень события из всех текущих ошибок контроллера в функциональном модуле «Motion Control» 0: ошибки контроллера отсутствуют. 2: частично критическая ошибка 3: некритическая ошибка	0, 2 или 3		
Code	Код события наивысшего уровня		Код события наиболее высокого уровня среди всех текущих ошибок контроллера в функциональном модуле «Motion Control» 16#0000_0000: ошибки контроллера отсутствуют. 16#0007_0000...16#FFFF_FFFF: код события	16#00000000, 16#00070000... 16#FFFFFFFF		

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	OK																			
Level							OK													
Code				OK																

Функция

Команда GetMCError определяет наиболее высокий уровень события и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «Motion Control» и выводит их в переменные *Level* и *Code*.
Если в данный момент ошибки контроллера отсутствуют, переменная *Out* (Флаг ошибки) будет содержать значение ЛОЖЬ.
Если имеется несколько ошибок контроллера с данным наиболее высоким уровнем события, в переменную *Code* записывается код события для ошибки контроллера, которая возникла первой.

Пример программы представлен на рисунке ниже.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_MC_ErrSta	Состояние ошибки управления движением	WORD	Содержит состояние ошибки функционального модуля «Motion Control». *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Пример программы

См. *Пример программы* на стр. 2-959 для команды ResetMCError.

ResetECError

Команда ResetECError сбрасывает ошибки контроллера в функциональном модуле «EtherCAT Master».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ResetECError	Сброс ошибки в EtherCAT	FB	ResetECError_instance ResetECError Execute Done Busy Error ErrorID	ResetECError_instance(Execute, Done, Busy, Error, ErrorID);

Переменные

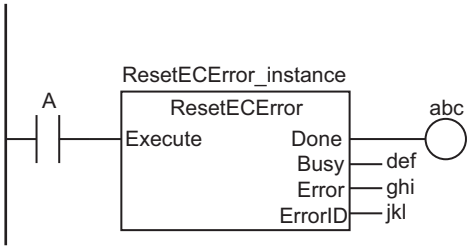
Используются только общие переменные.

Функция

Команда ResetECError сбрасывает ошибки контроллера в функциональном модуле «EtherCAT Master».

Пример программы представлен на рисунке ниже.

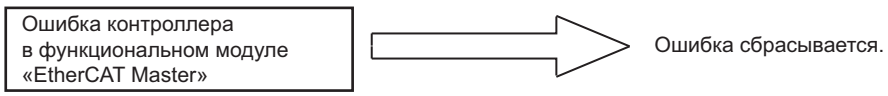
LD



ST

```
ResetECError_instance(A, abc, def, ghi, jkl);
```

Команда ResetECError сбрасывает ошибку контроллера в функциональном модуле «EtherCAT Master».



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_ErrSta	Ошибка встроенного интерфейса EtherCAT	WORD	Содержит сводную информацию об ошибках в функциональном модуле «EtherCAT Master». *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- После выполнения этой команды ошибка может быть сброшена не сразу, а с некоторой задержкой. Используйте команду GetECError, чтобы убедиться, что ошибки были сброшены.
- Если команда ResetECError будет выполнена, когда сеть находится в состоянии резервирования кабельных соединений, это состояние может быть временно сброшено.
- Если эту команду нужно выполнить для сервопривода OMRON серии G5, используйте эксклюзивное управления для команд, чтобы исключить одновременное выполнение команды ResetMCError, MC_Reset или MC_GroupReset с этой командой. Если любая из этих трех команд будет выполнена одновременно с командой ResetECError, сервопривод серии G5 больше не будет принимать сообщения SDO.
- Эту команду невозможно выполнить во время выполнения следующих команд: EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, ResetECError, RestartNXUnit и NX_ChangeWriteMode.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Эта команда выполняется повторно, когда еще идет активированный ранее процесс очистки ошибок контроллера в функциональном модуле «EtherCAT Master».
  b) Уже выполняется команда EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, ResetECError, RestartNXUnit или NX_ChangeWriteMode.

Пример программы

См. *Пример программы* на стр. 2-959 для команды ResetMCError.

GetECError

Команда GetECError обнаруживает ошибки в функциональном модуле «EtherCAT Master».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetECError	Получить состояние ошибки в EtherCAT	FUN	(@)GetECError EN — Out Level Code	Out:=GetECError(Level, Code);

Информация о версии

Для обнаружения ошибок ведомых устройств с помощью этой команды требуется модуль ЦПУ с версией модуля не ниже 1.02.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Флаг ошибки	Выход	ИСТИНА: имеется ошибка*1 ЛОЖЬ: ошибок нет	Зависит от типа данных.	---	---
Level	Наивысший уровень события		Наиболее высокий уровень события среди текущих ошибок*1 0: ошибок нет 2: частично критическая ошибка 3: некритическая ошибка	0, 2 или 3		
Code	Код события наивысшего уровня		Код события наиболее высокого уровня среди текущих ошибок*1	16#00000000, 16#00070000... 16#FFFFFFFF		

*1. Обнаруживаемые ошибки зависят от версии модуля ЦПУ и версии Sysmac Studio. Дополнительные сведения см. в разделе *Обнаруживаемые ошибки и значения выходных переменных* на стр. 2-969.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	ОК																			
Level							ОК													
Code				ОК																

Функция

Команда GetECError обнаруживает ошибки в функциональном модуле «EtherCAT Master».
Значение переменной *Out* равно ИСТИНА при наличии ошибки и ЛОЖЬ — при отсутствии ошибок.
Level передает состояние текущей ошибки наиболее высокого уровня.
Code передает код события текущей ошибки наиболее высокого уровня.

Обнаруживаемые ошибки и значения выходных переменных

Типы ошибок, которые может обнаружить данная команда, зависят от версии модуля ЦПУ. В следующей таблице для каждой версии модуля перечислены ошибки, которые могут быть обнаружены.

Версия модуля ЦПУ	Обнаруживаемые ошибки
1.02 или более поздняя версия	Ошибки портов связи, ошибки ведущего устройства и ошибки ведомых устройств
1.01 или более ранняя версия	Ошибки портов связи и ошибки ведущего устройства

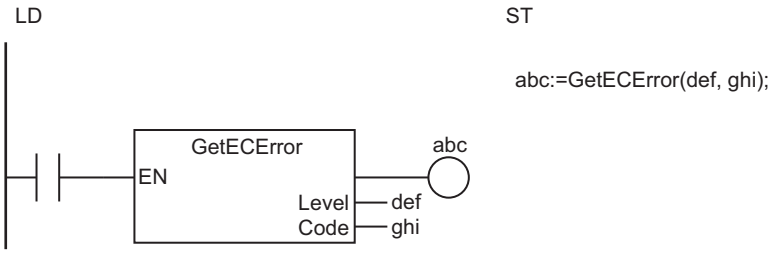
В следующей таблице отражена взаимосвязь между версией модуля ЦПУ, состоянием функционального модуля «EtherCAT Master» и значениями выходных переменных.

Версия модуля ЦПУ	Состояние функционального модуля «EtherCAT Master»	Значение *Out*	Значение *Level*	Значение *Code*
1.04 или более поздняя версия	Ошибок нет	ЛОЖЬ	0	16#0000_0000
	Ошибка порта связи или ошибка ведущего устройства	ИСТИНА	Наиболее высокий уровень события среди текущих ошибок 2: частично критическая ошибка 3: некритическая ошибка	Код события наиболее высокого уровня среди текущих ошибок*1 16#0007_0000...16#FFFF_FFFF
	Ошибка ведомого устройства			16#0000_0000
1.02 или 1.03	Ошибок нет	ЛОЖЬ	0	16#0000_0000
	Ошибка порта связи или ошибка ведущего устройства	ИСТИНА	Наиболее высокий уровень события среди текущих ошибок 2: частично критическая ошибка 3: некритическая ошибка	Код события наиболее высокого уровня среди текущих ошибок*1 16#0007_0000...16#FFFF_FFFF
	Ошибка ведомого устройства	ЛОЖЬ		16#0000_0000
1.00 или 1.01	Ошибок нет	ЛОЖЬ	0	16#0000_0000
	Ошибка порта связи или ошибка ведущего устройства	ИСТИНА	Наиболее высокий уровень события среди текущих ошибок 2: частично критическая ошибка 3: некритическая ошибка	Код события наиболее высокого уровня среди текущих ошибок*1 16#0007_0000...16#FFFF_FFFF
	Ошибка ведомого устройства	ЛОЖЬ	0	16#0000_0000

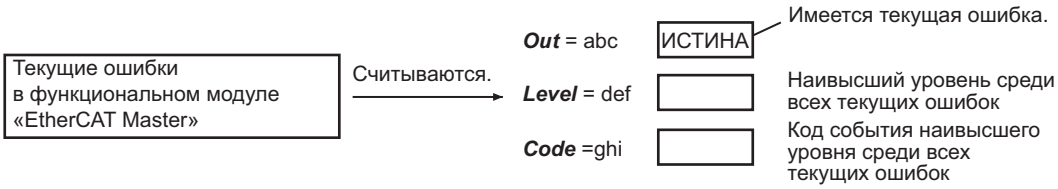
*1. Если имеется несколько ошибок с данным наиболее высоким уровнем события, указывается код события для ошибки, которая произошла первой.

Пример записи

Пример программы представлен на рисунке ниже.



Команда GetECError обнаруживает текущие ошибки портов связи, ошибки ведущего устройства и ошибки ведомых устройств в функциональном модуле «EtherCAT Master».



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_ErrSta	Ошибка встроенного интерфейса EtherCAT	WORD	Содержит сводную информацию об ошибках в функциональном модуле «EtherCAT Master». *1
_EC_PortErr *2	Ошибка порта связи	WORD	Содержит сводную информацию об ошибках портов связи ведущего устройства EtherCAT. *1
_EC_MstrErr *2	Ошибка ведущего устройства	WORD	Содержит сводную информацию об ошибках ведущего устройства EtherCAT и об ошибках ведомых устройств, обнаруженных ведущим устройством EtherCAT. *1
_EC_SlavErr	Ошибка ведомого устройства	WORD	Содержит сводную информацию о состоянии ошибок ведомых устройств EtherCAT в целом. *1

*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя (Cat. No. W505)*.

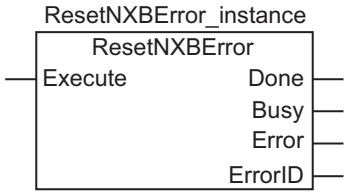
*2. Команда GetECError возвращает ошибки, которые указываются системными переменными _EC_PortErr (Ошибка порта связи) и _EC_MstrErr (Ошибка ведущего устройства).

Пример программы

См. *Пример программы* на стр. 2-959 для команды ResetMCError.

ResetNXBError

Команда ResetNXBError сбрасывает ошибки контроллера в функциональном модуле «NX Bus».

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ResetNXBError	Сброс ошибки в NX Bus	FB		ResetNXBError_instance(Execute, Done, Busy, Error, ErrorID);

Меры предосторожности для обеспечения надлежащей эксплуатации

Эту команду можно использовать для модулей ЦПУ NX102 и NX1P2.

Переменные

Используются только общие переменные.

Функция

Команда ResetNXBError сбрасывает текущие ошибки контроллера в функциональном модуле «NX Bus».

После сброса ошибки значение выходной переменной *Busy* меняется на ЛОЖЬ, а значение выходной переменной *Done* меняется на ИСТИНА.
Однако для модулей обеспечения безопасности ошибки не сбрасываются.

Если эта команда будет выполнена снова в экземпляре, отличном от экземпляра, для которого в данный момент уже выполняется сброс ошибок, последняя команда приведет к ошибке.
Выходная переменная *Error* этой последней команды перейдет в состояние ИСТИНА, а в выходную переменную *ErrorID* будет записан код ошибки (Многократное выполнение команд: 041A).

Связанные системные переменные

Имя	Значение
_NXB_ErrSta	Состояние ошибки функционального модуля «NX Bus»

Дополнительная информация

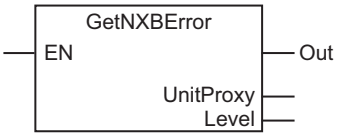
- При выполнении этой команды в средстве моделирования (Simulator) значение *Done* меняется на ИСТИНА, значения *Busy* и *Error* меняются на ЛОЖЬ, а значение *ErrorID* меняется на *0*, когда вход *Execute* переходит из состояния ЛОЖЬ в состояние ИСТИНА. Очистки ошибок при этом не происходит.
- Если причина ошибки не устранена и состояние ошибки после сброса ошибки остается прежним, сброс ошибки может оказаться невозможным.
- Журнал событий не очищается.

Меры предосторожности для обеспечения надлежащей эксплуатации

Эту команду невозможно использовать в событийной задаче. В этом случае произойдет ошибка компиляции.

GetNXBError

Команда GetNXBError определяет наиболее высокий уровень события среди текущих ошибок контроллера в функциональном модуле «NX Bus» модуля ЦПУ серии NX.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetNXBError	Получить состояние ошибки в NX Bus	FUN		Out:=GetNXBError(UnitProxy, Level);

Меры предосторожности для обеспечения надлежащей эксплуатации

Эту команду можно использовать для модулей ЦПУ NX102 и NX1P2.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Флаг ошибки	Выход	ИСТИНА: имеется ошибка ЛОЖЬ: ошибок нет	Зависит от типа данных.	---	---
UnitProxy	Модуль NX с наименьшим номером с ошибкой данного уровня		Модуль NX, в котором состояние текущий ошибки равно *Level*	---		
Level	Наивысший уровень события		Наиболее высокий уровень события среди текущих ошибок 0: ошибок нет 2: частично критическая ошибка 3: некритическая ошибка	0, 2 или 3		

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	OK																			
UnitProxy	Подробные сведения о структуре _sNXUNIT_ID см. в разделе *Функция* на стр. 2-974.																			
Level							OK													

Функция

Команда GetNXBError определяет наиболее высокий уровень события (частично критическая ошибка или некритическая ошибка) среди текущих ошибок контроллера в функциональном модуле «NX Bus».
Данная команда применяется к модулю ЦПУ NX102, модулю ЦПУ NX1P2 и модулю NX, подключенным к шине NX модуля ЦПУ.

Значение переменной *Out* равно ИСТИНА при наличии ошибки и ЛОЖЬ — при отсутствии ошибок.
Level передает состояние текущей ошибки наиболее высокого уровня.
Переменная *UnitProxy* возвращает значение *UnitProxy* модуля NX, в котором имеется текущая ошибка уровня *Level*.
Если ошибки одного уровня возникли сразу в нескольких модулях, она возвращает значение *UnitProxy* того модуля NX, который находится ближе всего к ведущему устройству и имеет наименьший номер модуля.

Для переменной *UnitProxy* используется структурный тип данных _sNXUNIT_ID. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных
UnitProxy	Модуль NX с наименьшим номером с ошибкой данного уровня	Модуль NX, в котором состояние текущий ошибки равно *Level*	_sNXUNIT_ID
NodeAdr	Адрес узла	Адрес узла интерфейсного модуля	UINT
IPAdr	IP-адрес	IP-адрес интерфейсного модуля	BYTE[5]
UnitNo	Номер модуля	Номер модуля NX	UDINT
Path	Путь	Информация о пути к модулю NX	BYTE[64]
PathLength	Допустимая длина пути *Path*	Допустимая длина пути *Path*	USINT

В *UnitProxy* должна передаваться переменная структурного типа _sNXUNIT_ID, созданная в таблице переменных.

Связанные системные переменные

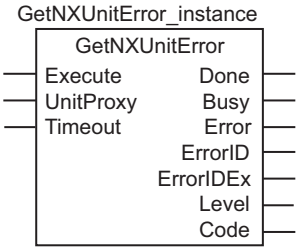
Имя	Значение
_NXB_ErrSta	Состояние ошибки функционального модуля «NX Bus»
_NXB_MstrErrSta	Состояние ошибки функционального модуля «NX Bus» как ведущего
_NXB_UnitErrSta[1]...[63]	Состояние ошибки модулей под управлением функционального модуля «NX Bus»

Дополнительная информация

При выполнении этой команды в средстве моделирования (Simulator) она всегда возвращает «No error» («Ошибок нет»), при этом *Out* меняется на ЛОЖЬ, *Level* меняется на *0*, а *UnitProxy* имеет неопределенное значение.

GetNXUnitError

Команда GetNXUnitError определяет наиболее высокий уровень события и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «NX Bus» модуля ЦПУ серии NX или модулей NX.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetNXUnitError	Получить состояние ошибки в модуле NX	FB		GetNXUnitError_instance(Execute , UnitProxy, TimeOut, Done, Busy, Error, ErrorID, ErrorIDEx, Level, Code);

Меры предосторожности для обеспечения надлежащей эксплуатации

Эту команду можно использовать для модулей ЦПУ NX102 и NX1P2.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitProxy	Указанный модуль	Вход	Указывает целевой модуль NX	---	---	*1
TimeOut	Время ожидания		Время ожидания Если задано значение 0, время ожидания составляет 2,0 с.	0...60000	мс	2000 (2,0 с)
Level	Наивысший уровень события	Выход	Наиболее высокий уровень события среди текущих ошибок 0: ошибок нет 2: частично критическая ошибка 3: некритическая ошибка	0, 2 или 3	---	---
Code	Код события наивысшего уровня		Код события наиболее высокого уровня среди всех текущих ошибок	16#00000000, 16#00070000... 16#FFFFFFFF		

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitProxy	Подробные сведения о структуре _sNXUNIT_ID см. в разделе *Функция* на стр. 2-976.																			
TimeOut							ОК													
Level							ОК													
Code				ОК																

Функция

Команда GetNXUnitError выдает наиболее высокий уровень события (частично критическая ошибка или некритическая ошибка) и код события наиболее высокого уровня среди текущих ошибок контроллера в функциональном модуле «NX Bus» и модулях NX для указанного модуля NX.
Данная команда применяется к модулю ЦПУ NX102, модулю ЦПУ NX1P2 и модулю NX, подключенным к шине NX модуля ЦПУ.

Модуль, из которого должны считываться данные, указывается параметром *UnitProxy*.
Выполнение команды завершается, когда значение переменной *Done* меняется на ИСТИНА.

Для переменной *UnitProxy* используется структурный тип данных _sNXUNIT_ID. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных
UnitProxy	Указанный модуль	Указанный модуль	_sNXUNIT_ID
NodeAdr	Адрес узла	Адрес узла интерфейсного модуля	UINT
IPAdr	IP-адрес	IP-адрес интерфейсного модуля	BYTE[5]
UnitNo	Номер модуля	Номер модуля указанного модуля	UDINT
Path	Путь	Информация о пути к указанному модулю	BYTE[64]
PathLength	Допустимая длина пути *Path*	Допустимая длина пути *Path*	USINT

В *UnitProxy* должна передаваться переменная устройства, назначенная указанному модулю.

Параметр *TimeOut* задает время ожидания. Если ответ не возвращается в течение времени ожидания, считается, что произошел сбой связи.

Level передает состояние текущей ошибки наиболее высокого уровня.
Code передает код события текущей ошибки наиболее высокого уровня. Если ошибки одного уровня возникли сразу в нескольких модулях, возвращается код события самой старой ошибки. При отсутствии ошибок возвращается значение 16#00000000.

Указанный модуль NX и значения переменных

Значения, выдаваемые во входные-выходные и выходные переменные, варьируются в зависимости от указанного модуля NX.

Взаимосвязь между указанным модулем NX и значением каждой переменной отражена в таблице ниже.

Указанный модуль	Модуль ЦПУ NX102 Модуль ЦПУ NX1P2	Модуль NX стойки модуля ЦПУ
Значение *Level*	Наивысший уровень события модуля ЦПУ	Наивысший уровень события модуля NX
Значение *Code*	Если ошибок несколько: код события наивысшего уровня Если есть несколько ошибок одного уровня (*Level*): код самого старого события Если ошибок нет: 16#0000_0000	

Применение совместно с командой GetNXBError

В программе пользователя обычно применяется команда GetNXBError (Получить состояние ошибки в NX Bus), которая проверяет, имеются ли ошибки в шине NX.

Если значение выходной переменной *Level* команды GetNXBError отличается от 0, в переменную *UnitProxy* этой команды записывается значение, указывающее модуль NX, в котором имеется ошибка с наиболее высоким уровнем события.

Для получения значений уровня (*Level*) и кода (*Code*) события для модуля NX, в котором произошла ошибка, необходимо выполнить команду GetNXUnitError.

Связанные системные переменные

Имя	Значение
_NXB_UnitErrFlagTbl	Состояние ошибки модуля NX

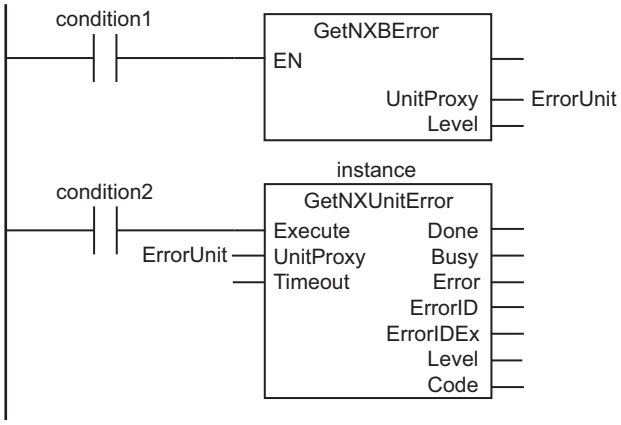
Дополнительная информация

Параметры, передаваемые в *UnitProxy*

Ниже поясняется, какие параметры должны быть переданы во входную переменную *UnitProxy*.

● Передача параметров в *UnitProxy* только с помощью программы пользователя

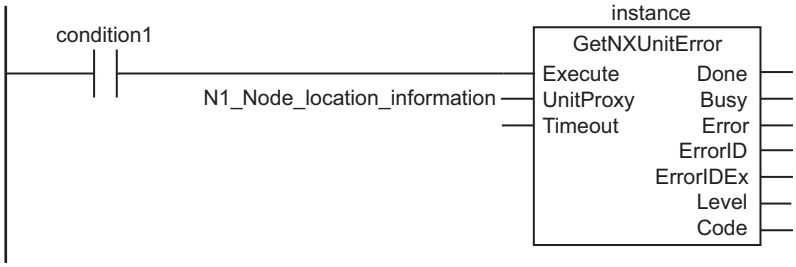
В переменную *UnitProxy* команды GetNXUnitError передается значение *ErrorUnit*. Оно содержит значение *UnitProxy*, возвращаемое командой GetNXBError, которая передается функциональному модулю «NX Bus».



● Передача параметров в *UnitProxy* с помощью переменной устройства

Создается переменная устройства, которая указывает модуль на шине NX. Эта переменная передается в переменную *UnitProxy* данной команды.

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	N1_Node_location_information	_sNXUNIT_ID	☑	



Выполнение в средстве моделирования (Simulator)

Когда выполняется эта команда и значение *Execute* меняется с ЛОЖЬ на ИСТИНА, значения связанных переменных изменяются указанным ниже образом.

Выходная переменная	Значение
Done	ИСТИНА
Busy	ЛОЖЬ
Error	ЛОЖЬ
ErrorID	0
ErrorIDEx	0
Level	0
Code	0

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- Не допускается выполнять несколько экземпляров данной команды одновременно. В любой момент времени может выполняться только один экземпляр команды.
- При возникновении ошибки состояние выхода *Error* меняется на «ИСТИНА». В следующей таблице приведены возможные значения переменных *ErrorID* и *ErrorIDEx* и поясняется их значение.

Значение *ErrorID*	Значение *ErrorIDEx*	Описание
16#0400	16#00000000	• Значение *UnitProxy* находится за пределами допустимого диапазона. • Значение *TimeOut* находится за пределами допустимого диапазона.
16#041A	16#00000000	Во время выполнения этой команды был выполнен другой экземпляр этой команды.
16#2C00	16#00000401	Указанный модуль не поддерживает команду.
	16#00001001 16#00001002 16#00170000 16#00200000 16#00210000	Неправильный входной, выходной или входной-выходной параметр. Проследите, чтобы для входного, выходного или входного-выходного параметра использовался предусмотренный параметр.
	16#00001010	Размер данных указанного объекта NX не согласуется с размером данных, который указан в параметре *WriteDat*.
	16#00001101	Неверный модуль. Проверьте модуль.
	16#0000110B	Размер прочитанных данных слишком велик. Убедитесь, что параметры читаемых данных верны и что читаемые данные указаны верно.
	16#00001110	Не существует объекта, соответствующего значению *Obj.Index*.
	16#00001111	Не существует объекта, соответствующего значению *Obj.Subindex*.
	16#00002101	Указанный объект NX не может быть записан.
	16#00002110	Значение *WriteDat* выходит за диапазон значений, которые могут быть записаны для объекта NX.
	16#00002210	Указанный модуль не находится в режиме, в котором разрешена запись данных.
	16#00002213	Выполнение команды было невозможно, поскольку указанный модуль выполнял проверку ввода-вывода. Выполните команду после завершения проверки ввода-вывода.

Значение *ErrorID*	Значение *ErrorIDEx*	Описание
	16#00002230	Состояние указанного модуля не согласуется со значением исходного объекта NX для чтения или целевого объекта NX для записи. Если значение *Obj.Index* находится в диапазоне от 0x6000 до 0x6FFF или от 0x7000 до 0x7FFF, выполните указанные ниже действия. • Удалите исходный объект NX для чтения или целевой объект NX для записи из параметров распределения данных ввода-вывода. • Сбросьте ошибку для указанного модуля. • Переведите указанный модуль в режим, в котором не разрешена запись данных.
	16#00002231	Выполнение команды было невозможно, поскольку указанный модуль выполнял инициализацию. Дождитесь, пока модуль начнет работать в обычном режиме, а затем выполните команду.
	16#0000250F	Не удалось получить доступ к оборудованию. Выполните команду еще раз.
	16#00002601 16#00002602 16#00100000	Указанный модуль не поддерживает эту команду. Проверьте версию модуля.
	16#00002603	Не удалось выполнить команду. Выполните команду еще раз. Убедитесь, что в параметрах выбора используемых каналов активирован (**Enabled**) хотя бы один канал.
	16#00002621	Модуль NX не находится в состоянии, в котором он может подтвердить команду. Подождите некоторое время, а затем выполните команду еще раз.
	16#00010000	Указанного модуля не существует. Убедитесь, что конфигурация модуля верна.
	16#00110000	Указанного номера порта не существует. Убедитесь, что конфигурация модуля верна.
	16#00120000 16#00130000 16#00150000 16#00160000	Неверное значение *UnitProxy*. Снова задайте переменную, которая указывает указанный интерфейсный модуль EtherCAT.
	16#00140000	Указан неправильный адрес узла. Убедитесь, что конфигурация модуля верна.
	16#00300000 16#80010000	Указанный модуль занят. Выполните команду еще раз.
	16#00310000	Для указанного модуля не поддерживается подключение. Проверьте версию модуля.
	16#80000000 16#80050000 16#81010000 16#81020000 16#82020000 16#82030000 16#82060000...16#8FFF0000 16#90010000...16#FFFE0000	Произошла ошибка в сети связи. Выполните команду еще раз.
	16#80020000 16#80030000 16#81030000 16#82000000	Произошла ошибка в сети связи. Уменьшите объем передаваемых данных.

Значение *ErrorID*	Значение *ErrorIDEx*	Описание
	16#80040000 16#81000000 16#82010000 16#82040000 16#82050000 16#90000000	Произошла ошибка в сети связи. Проверьте модуль и кабельные соединения. Убедитесь, что включен источник питания модуля.
16#2C02	16#00000000	Превышено время ожидания при обмене данными.

Пример программы

В качестве примера рассмотрим программу, которая передает значения модуля, уровня и кода в соответствующие переменные для отображения на сенсорной панели в случае возникновения ошибки в шине NX.

Система имеет следующую конфигурацию:
три модуля NX подключены к модулю ЦПУ NX1P2.

Переменные устройств

Модули NX	Переменная устройства
1-й модуль NX	N1_Node_location_information
2-й модуль NX	N2_Node_location_information
3-й модуль NX	N3_Node_location_information

Определения глобальных переменных

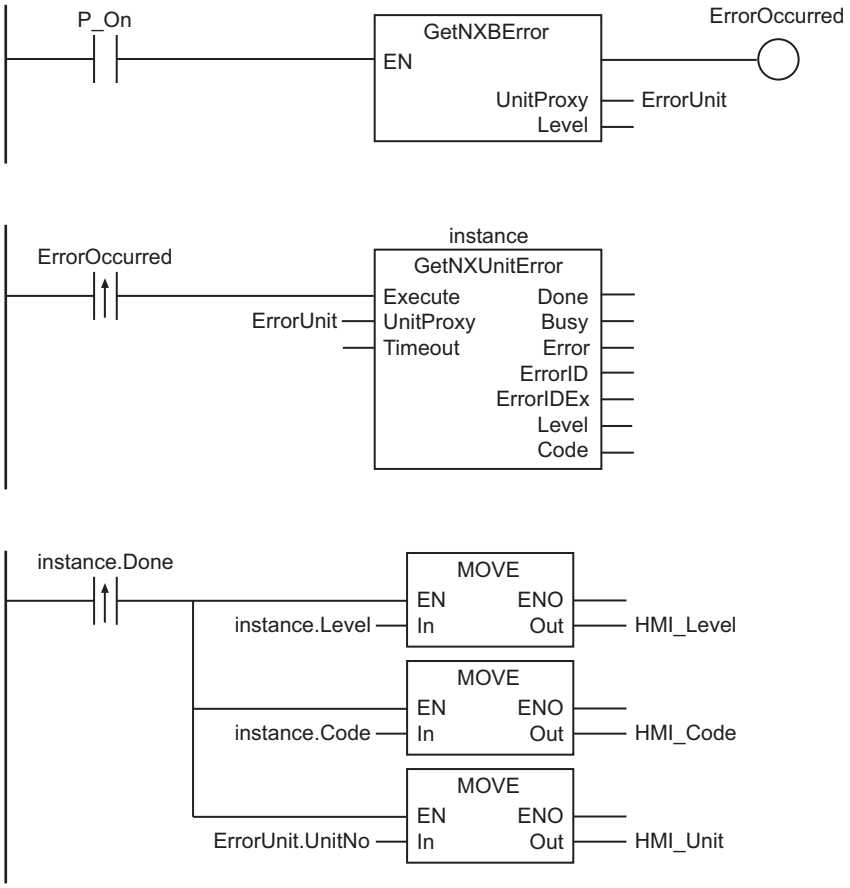
● Глобальные переменные

Переменная	Тип данных	Константа	Комментарий
N1_Node_location_information	_sNXUNIT_ID	☑	
N2_Node_location_information	_sNXUNIT_ID	☑	
N3_Node_location_information	_sNXUNIT_ID	☑	
HMI_Level *1	UINT	☐	
HMI_Code *1	DWORD	☐	
HMI_Unit *1	UDINT	☐	

*1. Переменные, имя которых начинается с *HMI_*, являются переменными для отображения на сенсорной панели.

Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Константа	Комментарий
	instance	GetNXUnitError	⌐¬	
	ErrorOccurred	BOOL	⌐¬	
	ErrorUnit	_sNXUNIT_ID	⌐¬	



SetInfo

Команда SetInfo создает информацию пользователя.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SetInfo	Создание информации пользователя	FUN	(@)SetInfo EN ENO Code Out Info1 Info2	SetInfo(Code, Info1, Info2);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Code	Код события	Вход	Код события генерируемой информации пользователя	40001...60000	---	40001
Info1	Прилагаемая информация 1		Значения, записываемые в журнал событий при активации информации пользователя	Зависит от типа данных.		*1
Info2	Прилагаемая информация 2					
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

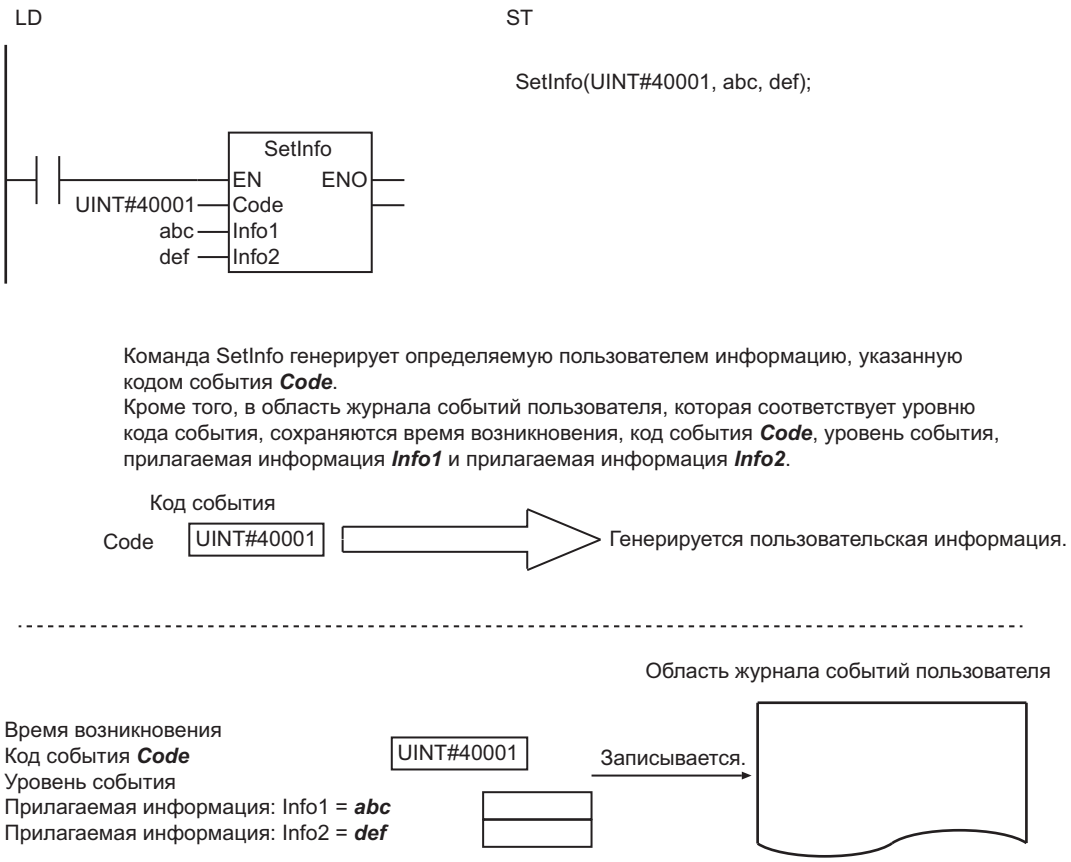
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Code							OK													
Info1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
Info2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
Out	OK																			

Функция

Команда SetInfo генерирует определенную пользователем информацию, которая указана кодом события *Code*.

В область журнала событий пользователя, которая соответствует уровню кода события, сохраняются следующие данные: время возникновения, код события *Code*, уровень события, прилагаемая информация *Info1* и прилагаемая информация *Info2*.

Пример программы представлен на рисунке ниже. Создается информация пользователя для кода события 40001. В качестве прилагаемой информации сохраняются значения переменных *abc* и *def*.

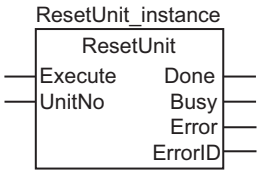


Меры предосторожности для обеспечения надлежащей эксплуатации

- Обязательно используйте переменные для входных параметров, передаваемых в *Info1* и *Info2*. Если прилагаемая информация не используется, укажите фиктивную переменную. Если будет указана константа, произойдет ошибка сборки.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанном ниже случае произойдет ошибка. Выход *ENO* перейдет в состояние ЛОЖЬ.
  а) Значение *Code* находится за пределами допустимого диапазона.

ResetUnit

Команда ResetUnit перезапускает модуль шины ЦПУ или специальный модуль ввода-вывода.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ResetUnit	Перезапустить модуль	FB		ResetUnit_instance(Execute, UnitNo, Done, Busy, Error, ErrorID);

Меры предосторожности для обеспечения надлежащей эксплуатации

Данную команду невозможно использовать с модулями ЦПУ серии NX.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitNo	Номер модуля	Вход	Номер модуля для перезапуска	_CBU_No00..._ CBU_No15, _SIO_No00..._ SIO_No95	---	_CBU_No00

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitNo	Сведения о перечислителях перечислимого типа _eUnitNo см. в разделе *Функция* на стр. 2-985.																			

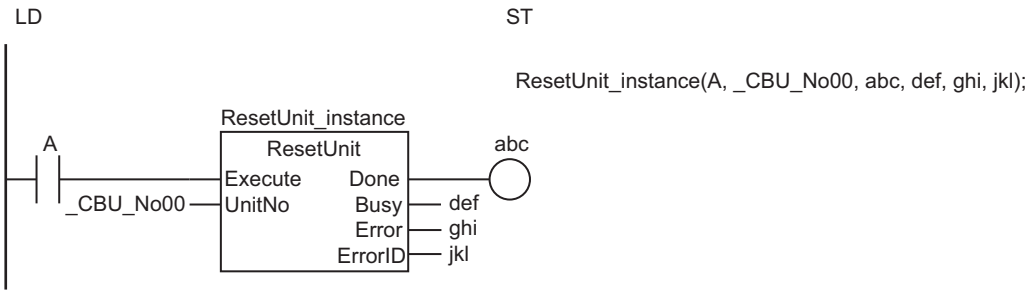
Функция

Команда ResetUnit перезапускает модуль шины ЦПУ или специальный модуль ввода-вывода. Перезапускаемый модуль указывается параметром *UnitNo*.

Для параметра *UnitNo* используется перечислимый тип данных _eUnitNo. Значения перечислителей приведены в таблице ниже.

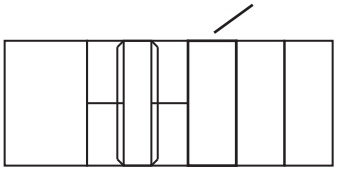
Перечислители	Значение
_CBU_No00..._CBU_No15	Номер модуля шины ЦПУ, 00...15
_SIO_No00..._SIO_No95	Номер модуля специального модуля ввода-вывода, 00...95

Ниже приведен пример для случая, когда *UnitNo* = _CBU_No00. Перезапускается модуль шины ЦПУ с номером модуля 0.



Команда ResetUnit перезапускает модуль шины ЦПУ или специальный модуль ввода-вывода с номером ***UnitNo***.

UnitNo = _CBU_No00: перезапускается модуль шины ЦПУ с номером модуля 0.



Меры предосторожности для обеспечения надлежащей эксплуатации

- Даже если для модуля, указанного параметром *UnitNo*, в данный момент уже выполняется операция перезапуска, эта команда не будет завершена с ошибкой. Переменная *Busy* сохранит значение ИСТИНА, а значение *Done* поменяется на ИСТИНА, когда завершится операция перезапуска. Запросы на перезапуск не ставятся в очередь.
- Если в начале работы на входе *Execute* присутствует состояние ИСТИНА, модуль перезапускается.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение *UnitNo* находится за пределами допустимого диапазона.
  b) Модуля с указанным номером *UnitNo* не существует.
  c) Не удалось выполнить перезапуск.

Пример программы

Когда значение *Trigger* меняется на ИСТИНА, для скорости передачи последовательного порта 1 модуля последовательного интерфейса с номером модуля 0 устанавливается значение 38 400 бит/с, после чего модуль перезапускается.

Определения глобальных переменных

● Глобальные переменные

Имя	Тип данных	Начальное значение	Параметр «AT»*1	Сохранение	Комментарий
SCU_P1_BaudrateCfg	USINT	0	IOBus://rack#0/slot#0/P1_BaudrateCfg	☑	Скорость передачи

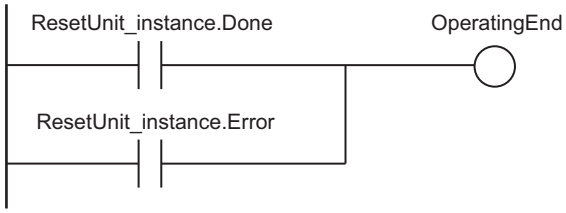
*1. Параметр «AT» для случая, когда модуль последовательного интерфейса установлен в слот с номером 0 стойки с номером 0.

Программа на языке LD

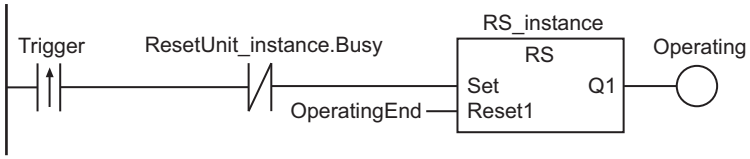
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	RS_instance	RS		
	ResetUnit_instance	ResetUnit		

Внешние переменные	Переменная	Тип данных	Комментарий
	SCU_P1_BaudrateCfg	USINT	Скорость передачи

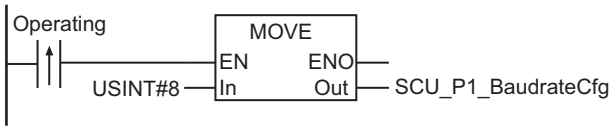
Определение, завершилось ли выполнение ResetUnit.



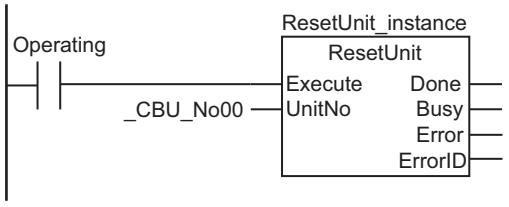
Прием условия выполнения.



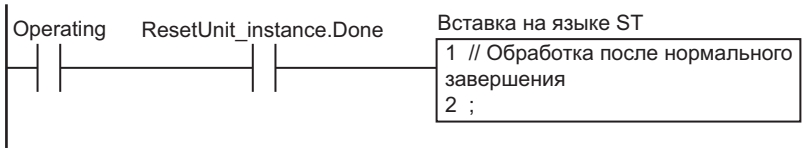
Установка скорости передачи в переменной устройства.



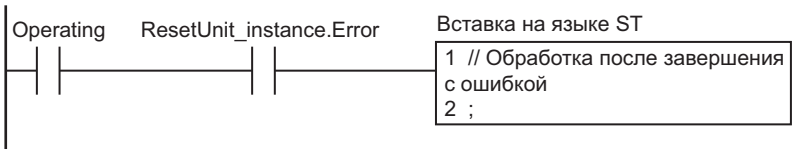
Выполнение команды ResetUnit.



Обработка после нормального завершения



Обработка после завершения с ошибкой



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась
	Operating	BOOL	ЛОЖЬ	Обработка
	ResetUnit_instance	ResetUnit		

Внешние переменные	Переменная	Тип данных	Комментарий
	SCU_P1_BaudrateCfg	USINT	Скорость передачи

```
// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (ResetUnit_instance.Busy=FALSE) ) T
HEN
  OperatingStart:=TRUE;
  Operating:=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация ResetUnit_instance и установка скорости передачи в переменной устр
ойства.
IF (OperatingStart=TRUE) THEN
  ResetUnit_instance(Execute:=FALSE);
  SCU_P1_BaudrateCfg:=USINT#8;
  OperatingStart:=FALSE;
END_IF;

// Выполнение команды ResetUnit.
IF (Operating=TRUE) THEN
  ResetUnit_instance(
    Execute:=TRUE, // Условие выполнения
    UnitNo :=_CBU_No00); // Номер модуля

  IF (ResetUnit_instance.Done=TRUE) THEN
    // Обработка после нормального завершения
```

```
    Operating:=FALSE;
  END_IF;

  IF (ResetUnit_instance.Error=TRUE) THEN
    // Обработка после завершения с ошибкой
    Operating:=FALSE;
  END_IF;
END_IF;
```

GetNTPStatus

Команда GetNTPStatus производит чтение состояния NTP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetNTPStatus	Чтение состояния NTP	FUN	(@)GetNTPStatus EN — ENO — Out ExecTime ExecNormal	GetNTPStatus(ExecTime, ExecNormal);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---
ExecTime	Время последней нормальной операции NTP		Время последней нормальной операции NTP	Зависит от типа данных.	Год, месяц, день, час, минуты, секунды	
ExecNormal	Флаг нормального завершения NTP		ИСТИНА: нормальное завершение ЛОЖЬ: завершение с ошибкой		---	

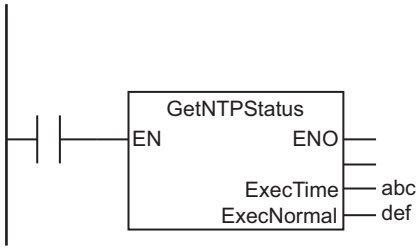
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	ОК																			
ExecTime																			ОК	
ExecNormal	ОК																			

Функция

Команда GetNTPStatus производит чтение состояния NTP.
Считывается следующая информация: *ExecTime* (Время последней нормальной операции NTP) и *ExecNormal* (Флаг нормального завершения NTP).

Пример программы представлен на рисунке ниже.

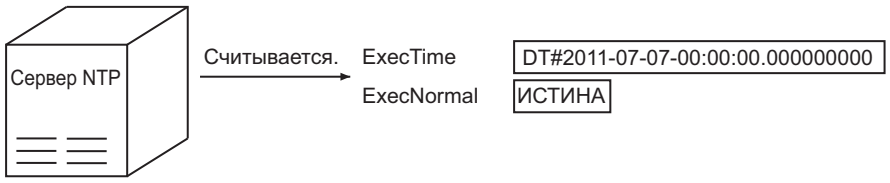
LD



ST

```
GetNTPStatus(abc, def);
```

Команда GetNTPStatus считывает состояние NTP.
Если нормальная работа NTP в последний раз наблюдалась в 00:00.00, 7 июля 2011 г., то ***ExecTime*** и ***ExecNormal*** будут содержать следующие значения.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_NTPResult	Состояние NTP	_sNTP_RESULT	Содержит состояние NTP.*1

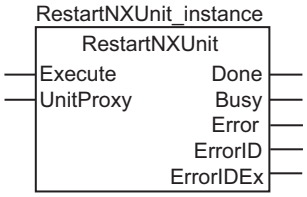
*1. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- При использовании этой команды в программе на языке ST возвращаемое значение *Out* не используется.
- Данная команда считывает содержимое системной переменной *_EIP_NTPResult*. К этой переменной невозможно обратиться напрямую. Для чтения содержимого этой переменной всегда следует использовать данную команду.

RestartNXUnit

Команда RestartNXUnit перезапускает интерфейсный модуль EtherCAT или модули NX.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
RestartNXUnit	Перезапуск модулей NX	FB		RestartNXUnit_instance(Execute, UnitProxy, Done, Busy, Error, ErrorID, ErrorIDEx);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.05 или более поздней и Sysmac Studio версии 1.06 или выше.
Однако некоторые версии/версии модуля указанных ниже продуктов не поддерживают функцию *перезапуска указанных модулей NX*.

- Модули ЦПУ
- Sysmac Studio
- Интерфейсные модули EtherCAT
- Модули NX

Если версия модуля продукта не поддерживает функцию *перезапуска указанных модулей NX*, в качестве модуля для перезапуска можно указать только интерфейсный модуль EtherCAT.
Документ *Серия NX, интерфейсный модуль EtherCAT — Руководство пользователя* (Cat. No. W519-E1-03 или более поздней редакции) содержит сведения о версиях модуля продуктов, поддерживающих функцию *перезапуска указанных модулей NX*.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitProxy	Указанный модуль	Вход	Перезапускаемый модуль: интерфейсный модуль EtherCAT, функциональный модуль «NX Bus» или модуль NX	---	---	*1

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitProxy	Подробные сведения о структуре _sNXUNIT_ID см. в разделе *Функция* на стр. 2-993.																			

Функция

Команда RestartNXUnit перезапускает интерфейсный модуль EtherCAT или модуль NX, подключенный к интерфейсному модулю EtherCAT, а также модуль NX, подключенный к шине NX функционального модуля «NX Bus» или модуля ЦПУ.
С помощью этой команды можно индивидуально перезапустить конкретный модуль.
Однако интерфейсный модуль EtherCAT или функциональный модуль «NX Bus» перезапустить индивидуально (независимо) невозможно.
Если указать интерфейсный модуль EtherCAT или функциональный модуль «NX Bus», также будут перезапущены все подключенные к нему модули NX.
Модуль, который должен быть перезапущен, указывается параметром *UnitProxy*.

Для переменной *UnitProxy* используется структурный тип данных _sNXUNIT_ID. Значения членов структуры приведены в таблице ниже.

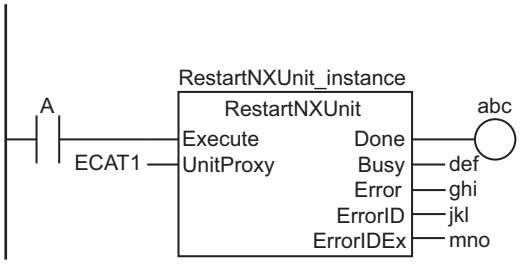
Имя		Значение	Содержание	Тип данных
UnitProxy		Указанный модуль	Указанный модуль	_sNXUNIT_ID
	NodeAdr	Адрес узла	Адрес узла интерфейсного модуля	UINT
	IPAdr	IP-адрес	IP-адрес интерфейсного модуля	BYTE[5]
	UnitNo	Номер модуля	Номер модуля указанного модуля	UDINT
	Path	Путь	Информация о пути к указанному модулю	BYTE[64]
	PathLength	Допустимая длина пути *Path*	Допустимая длина пути *Path*	USINT

В *UnitProxy* должна быть передана переменная устройства, назначенная указанному интерфейсному модулю EtherCAT или модулю NX, подключенному к интерфейсному модулю EtherCAT, либо модулю NX, подключенному к шине NX функционального модуля «NX Bus» или модуля ЦПУ.

Пример записи

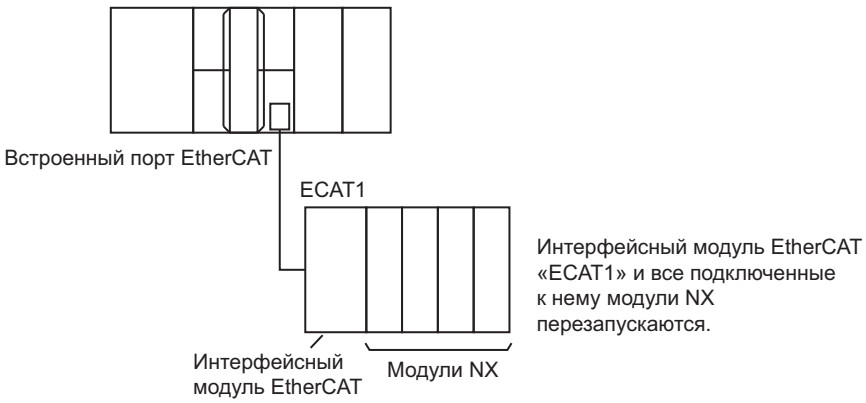
В представленном ниже примере перезапускаются все ведомые терминалы EtherCAT. Интерфейсному модулю EtherCAT назначается переменная с именем *ECAT1* и типом данных _sNXUNIT_ID.

LD



ST

```
RestartNXUnit_instance(A, ECAT1,
                       abc, def, ghi, jkl, mno);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_MBXSlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, способных участвовать в обмене сообщениями	BOOL	Эта переменная указывает возможность связи для каждого ведомого устройства. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_NXB_UnitMsgActiveTbl[i]	Состояние активации обмена сообщениями модуля NX	BOOL	В этой таблице указываются ведомые устройства, которые способны участвовать в обмене сообщениями. С помощью этой переменной можно проверить, возможен ли обмен сообщениями с конкретным ведомым устройством.

Дополнительная информация

Ниже описана процедура, с помощью которой можно записать данные с указанными ниже атрибутами в интерфейсный модуль EtherCAT, модуль NX, подключенный к интерфейсному модулю EtherCAT, или модуль NX, подключенный к шине NX модуля ЦПУ.

- Атрибут хранения данных при выключенном питании
- Значения обновляются при перезапуске модуля.

1 С помощью команды *NX_ChangeWriteMode* на стр. 2-998 переведите модуль в режим, допускающий запись данных.

2 Используйте команду *NX_WriteObj* на стр. 2-1114 для записи данных в модуль.

3 Используйте команду *NX_SaveParam* на стр. 2-1004 для сохранения записанных данных.

4 С помощью команды RestartNXUnit перезапустите модуль.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если в параметре *UnitProxy* указан модуль, который назначен для оси управления движением (тип данных _sAXIS_REF), в функциональном модуле «Motion Control» произойдет ошибка контроллера. Команда *ResetMCError* на стр. 2-957 служит для сброса ошибки контроллера.
- Для *UnitProxy* должна быть указана переменная устройства, которая назначена [интерфейсному модулю EtherCAT или модулю NX, подключенному к интерфейсному модулю EtherCAT, либо модулю NX, подключенному к шине NX функционального модуля «NX Bus» или модуля ЦПУ] на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
  Информацию о назначении переменных устройства см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.
- При попытке выполнить команду RestartNXUnit во время выполнения другой команды RestartNXUnit или команды *NX_ChangeWriteMode* на стр. 2-998 она будет поставлена в очередь. В очередь может быть поставлено до 192 команд. При попытке поставить в очередь более 192 команд произойдет ошибка сборки. Время, в течение которого команда находится в очереди, не включается в значение времени ожидания.
- Пока команда находится в очереди, выход *Busy* находится в состоянии ИСТИНА.
- Эта команда имеет отношение к ошибкам обмена сообщениями NX. Если одновременно будет выполнено слишком много команд, имеющих отношение к ошибкам обмена сообщениями NX, произойдет ошибка обмена сообщениями NX. Список команд, имеющих отношение к ошибкам обмена сообщениями NX, см. в разделе *A-4 Команды, связанные с ошибками обмена сообщениями NX* на стр. A-38.
- Эту команду невозможно выполнить во время выполнения следующих команд: EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, ResetECError, RestartNXUnit и NX_ChangeWriteMode. При попытке ее выполнить произойдет ошибка.
- При возникновении ошибки состояние выхода *Error* поменяется на ИСТИНА. В следующей таблице приведены возможные значения переменных *ErrorID* и *ErrorIDEx* и поясняется их значение.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
16#0419	16#00000000	Неправильный тип данных *UnitProxy*.
16#2C00	16#00000401	Указанный модуль не поддерживает команду.
	16#00001001 16#00001002 16#00170000 16#00200000 16#00210000	Неправильный входной, выходной или входной-выходной параметр. Проследите, чтобы для входного, выходного или входного-выходного параметра использовался предусмотренный параметр.
	16#00001010	Размер данных указанного объекта NX не согласуется с размером данных, который указан в параметре *WriteDat*.
	16#00001101	Неверный модуль. Проверьте модуль.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
	16#0000110B	Размер прочитанных данных слишком велик. Убедитесь, что параметры читаемых данных верны и что читаемые данные указаны верно.
	16#00001110	Не существует объекта, соответствующего значению *Obj.Index*.
	16#00001111	Не существует объекта, соответствующего значению *Obj.Subindex*.
	16#00002101	Указанный объект NX не может быть записан.
	16#00002110	Значение *WriteDat* выходит за диапазон значений, которые могут быть записаны для объекта NX.
	16#00002210	Указанный модуль не находится в режиме, в котором разрешена запись данных.
	16#00002213	Выполнение команды было невозможно, поскольку указанный модуль выполнял проверку ввода-вывода. Выполните команду после завершения проверки ввода-вывода.
	16#00002230	Состояние указанного модуля не согласуется со значением исходного объекта NX для чтения или целевого объекта NX для записи. Если значение *Obj.Index* находится в диапазоне от 0x6000 до 0x6FFF или от 0x7000 до 0x7FFF, выполните указанные ниже действия. • Удалите исходный объект NX для чтения или целевой объект NX для записи из параметров распределения данных ввода-вывода. • Сбросьте ошибку для указанного модуля. • Переведите указанный модуль в режим, в котором не разрешена запись данных.
	16#00002231	Выполнение команды было невозможно, поскольку указанный модуль выполнял инициализацию. Дождитесь, пока модуль начнет работать в обычном режиме, а затем выполните команду.
	16#0000250F	Не удалось получить доступ к оборудованию. Выполните команду еще раз.
	16#00002601 16#00002602 16#00100000	Указанный модуль не поддерживает эту команду. Проверьте версию модуля.
	16#00002603	Не удалось выполнить команду. Выполните команду еще раз. Убедитесь, что в параметрах выбора используемых каналов активирован (**Enabled**) хотя бы один канал.
	16#00002621	Модуль NX не находится в состоянии, в котором он может подтвердить команду. Подождите некоторое время, а затем выполните команду еще раз.
	16#00010000	Указанного модуля не существует. Убедитесь, что конфигурация модуля верна.
	16#00110000	Указанного номера порта не существует. Убедитесь, что конфигурация модуля верна.
	16#00120000 16#00130000 16#00150000 16#00160000	Неверное значение *UnitProxy*. Снова задайте переменную, которая указывает указанный интерфейсный модуль EtherCAT.
	16#00140000	Указан неправильный адрес узла. Убедитесь, что конфигурация модуля верна.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
	16#00300000 16#80010000	Указанный модуль занят. Выполните команду еще раз.
	16#00310000	Для указанного модуля не поддерживается подключение. Проверьте версию модуля.
	16#80000000 16#80050000 16#81010000 16#81020000 16#82020000 16#82030000 16#82060000...16#8FFF0000 16#90010000...16#FFFE0000	Произошла ошибка в сети связи. Выполните команду еще раз.
	16#80020000 16#80030000 16#81030000 16#82000000	Произошла ошибка в сети связи. Уменьшите объем передаваемых данных.
	16#80040000 16#81000000 16#82010000 16#82040000 16#82050000 16#90000000	Произошла ошибка в сети связи. Проверьте модуль и кабельные соединения. Убедитесь, что включен источник питания модуля.
16#2C01	16#00000000	Была предпринята попытка поставить в очередь более 192 команд RestartNXUnit и NX_ChangeWriteMode.
16#2C02	16#00000000	Превышено время ожидания при обмене данными.
16#2C05	---	Произошла ошибка в сети EtherCAT. Проверьте значение *UnitProxy* и конфигурацию сети EtherCAT.
16#2C06	16#00000000	Указанный модуль в данный момент уже перезапускается из Sysmac Studio. Поэтому эту команду выполнять не требуется.
16#2C07	16#00000000	По адресу ведомого узла указанного модуля было подключено ведомое устройство, которое не может быть указано для команды. Проверьте значение *UnitProxy* и конфигурацию сети EtherCAT.

Пример программы

См. *Пример программы* на стр. 2-1119 для команды NX_WriteObj.

NX_ChangeWriteMode

Команда NX_ChangeWriteMode переводит интерфейсный модуль EtherCAT или модуль NX в режим, в котором разрешена запись данных.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_ChangeWriteMode	Перевод модуля NX в режим записи	FB	NX_ChangeWriteMode_instance NX_ChangeWriteMode Execute UnitProxy Done Busy Error ErrorID ErrorIDEx	NX_ChangeWriteMode_instance(Execute, UnitProxy, Done, Busy, Error, ErrorID, ErrorIDEx);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.05 или более поздней и Sysmac Studio версии 1.06 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitProxy	Указанный модуль	Вход	Модуль, режим работы которого нужно изменить	---	---	*1

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitProxy	Подробные сведения о структуре _sNXUNIT_ID см. в разделе *Функция* на стр. 2-998.																			

Функция

Команда NX_ChangeWriteMode переводит интерфейсный модуль EtherCAT, модуль NX, подключенный к интерфейсному модулю EtherCAT, или модуль NX, подключенный к шине NX модуля ЦПУ, в режим, в котором в модуль могут быть записаны данные.
Модуль, режим работы которого нужно изменить, указывается параметром *UnitProxy*.
Запись данных становится возможной после того, как значение переменной *Done* меняется на ИСТИНА.

Для переменной *UnitProxy* используется структурный тип данных _sNXUNIT_ID. Значения членов структуры приведены в таблице ниже.

Имя		Значение	Содержание	Тип данных
UnitProxy		Указанный модуль	Модуль, для которого нужно изменить режим записи	_sNXUNIT_ID
	NodeAdr	Адрес узла	Адрес узла интерфейсного модуля	UINT
	IPAdr	IP-адрес	IP-адрес интерфейсного модуля	BYTE[5]
	UnitNo	Номер модуля	Номер модуля указанного модуля	UDINT
	Path	Путь	Информация о пути к указанному модулю	BYTE[64]
	PathLength	Допустимая длина пути *Path*	Допустимая длина пути *Path*	USINT

В *UnitProxy* следует передать переменную устройства, которая назначена указанному модулю.

Связанные команды и порядок выполнения

С помощью этой команды можно записать данные с указанными ниже атрибутами в интерфейсный модуль EtherCAT, модуль NX, подключенный к интерфейсному модулю EtherCAT, или модуль NX, подключенный к шине NX модуля ЦПУ.

- Атрибут хранения данных при выключенном питании
- Значения обновляются при перезапуске модуля.

Для выполнения связанных команд соблюдайте следующий порядок действий.

1 С помощью команды NX_ChangeWriteMode переведите модули в режим, допускающий запись данных.

2 Используйте команду *NX_WriteObj* на стр. 2-1114 для записи данных в модуль.

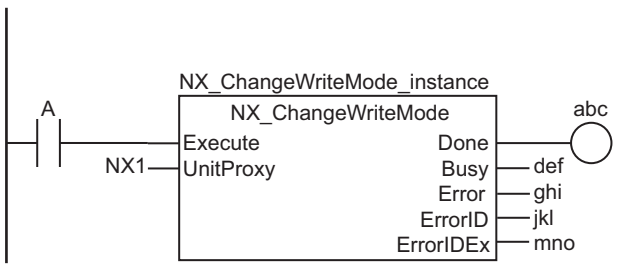
3 Используйте команду *NX_SaveParam* на стр. 2-1004 для сохранения записанных данных.

4 С помощью команды *RestartNXUnit* на стр. 2-992 перезапустите модуль.

Пример записи

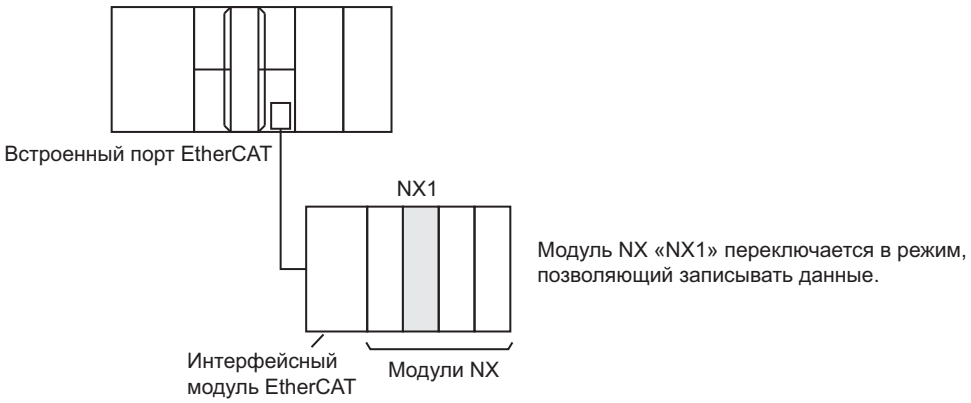
Ниже показан пример записи команды для перевода модуля NX (*NX1*) в режим, в котором разрешена запись данных. Модулю NX, режим работы которого изменяется, назначена переменная с именем *NX1* и типом данных _sNXUNIT_ID.

LD



ST

```
NX_ChangeWriteMode_instance(A, NX1, abc, def,
                            ghi, jkl, mno);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_MBXSlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, способных участвовать в обмене сообщениями	BOOL	Эта переменная указывает возможность связи для каждого ведомого устройства. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_NXB_UnitMsgActiveTbl[i]	Состояние активации обмена сообщениями модуля NX	BOOL	В этой таблице указываются ведомые устройства, которые способны участвовать в обмене сообщениями. С помощью этой переменной можно проверить, возможен ли обмен сообщениями с конкретным ведомым устройством.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если в параметре *UnitProxy* указан модуль, который назначен для оси управления движением (тип данных _sAXIS_REF), в функциональном модуле «Motion Control» произойдет ошибка контроллера. В этом случае используйте команду *ResetMCError* на стр. 2-957 для сброса ошибки контроллера.
- Для *UnitProxy* должна быть указана переменная устройства, которая назначена интерфейсному модулю EtherCAT; модулю NX, подключенному к интерфейсному модулю EtherCAT; или модулю NX, подключенному к шине NX модуля ЦПУ, на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
  Информацию о назначении переменных устройства см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.
- При попытке выполнить команду NX_ChangeWriteMode во время выполнения другой команды NX_ChangeWriteMode или команды *RestartNXUnit* на стр. 2-992 она будет поставлена в очередь. В очередь может быть поставлено до 192 команд. При попытке поставить в очередь более 192 команд произойдет ошибка сборки. Время, в течение которого команда находится в очереди, не включается в значение времени ожидания.

- Пока команда находится в очереди, выход *Busy* находится в состоянии ИСТИНА.
- Эта команда имеет отношение к ошибкам обмена сообщениями NX. Если одновременно будет выполнено слишком много команд, имеющих отношение к ошибкам обмена сообщениями NX, произойдет ошибка обмена сообщениями NX. Список команд, имеющих отношение к ошибкам обмена сообщениями NX, см. в разделе *A-4 Команды, связанные с ошибками обмена сообщениями NX* на стр. A-38.
- Эту команду невозможно выполнить во время выполнения следующих команд: EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, ResetECError, RestartNXUnit и NX_ChangeWriteMode. При попытке ее выполнить произойдет ошибка.
- При возникновении ошибки состояние выхода *Error* поменяется на ИСТИНА. В следующей таблице приведены возможные значения переменных *ErrorID* и *ErrorIDEx* и поясняется их значение.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
16#0419	16#00000000	Неправильный тип данных *UnitProxy*.
16#2C00	16#00000401	Указанный модуль не поддерживает команду.
	16#00001001 16#00001002 16#00170000 16#00200000 16#00210000	Неправильный входной, выходной или входной-выходной параметр. Проследите, чтобы для входного, выходного или входного-выходного параметра использовался предусмотренный параметр.
	16#00001010	Размер данных указанного объекта NX не согласуется с размером данных, который указан в параметре *WriteDat*.
	16#00001101	Не указан правильный модуль. Проверьте модуль.
	16#0000110B	Размер прочитанных данных слишком велик. Убедитесь, что параметры читаемых данных верны и что читаемые данные указаны верно.
	16#00001110	Не существует объекта, соответствующего значению *Obj.Index*.
	16#00001111	Не существует объекта, соответствующего значению *Obj.Subindex*.
	16#00002101	Указанный объект NX не может быть записан.
	16#00002110	Значение *WriteDat* выходит за диапазон значений, которые могут быть записаны для объекта NX.
	16#00002210	Указанный модуль не находится в режиме, в котором разрешена запись данных.
	16#00002213	Выполнение команды было невозможно, поскольку указанный модуль выполнял проверку ввода-вывода. Выполните команду после завершения проверки ввода-вывода.
	16#00002230	Состояние указанного модуля не согласуется со значением исходного объекта NX для чтения или целевого объекта NX для записи. Если значение *Obj.Index* находится в диапазоне от 0x6000 до 0x6FFF или от 0x7000 до 0x7FFF, выполните указанные ниже действия. • Удалите исходный объект NX для чтения или целевой объект NX для записи из параметров распределения данных ввода-вывода. • Сбросьте ошибку для указанного модуля. • Переведите указанный модуль в режим, в котором не разрешена запись данных.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
	16#00002231	Выполнение команды было невозможно, поскольку указанный модуль выполнял инициализацию. Дождитесь, пока модуль начнет работать в обычном режиме, а затем выполните команду.
	16#0000250F	Не удалось получить доступ к оборудованию. Выполните команду еще раз.
	16#00002601 16#00002602 16#00100000	Указанный модуль не поддерживает эту команду. Проверьте версию модуля.
	16#00002603	Не удалось выполнить команду. Выполните команду еще раз. Убедитесь, что в параметрах выбора используемых каналов активирован (**Enabled**) хотя бы один канал.
	16#00002621	Модуль NX не находится в состоянии, в котором он может подтвердить команду. Подождите некоторое время, а затем выполните команду еще раз.
	16#00010000	Указанного модуля не существует. Убедитесь, что конфигурация модуля верна.
	16#00110000	Указанного номера порта не существует. Убедитесь, что конфигурация модуля верна.
	16#00120000 16#00130000 16#00150000 16#00160000	Неверное значение *UnitProxy*. Снова задайте переменную, которая указывает указанный интерфейсный модуль EtherCAT.
	16#00140000	Указан неправильный адрес узла. Убедитесь, что конфигурация модуля верна.
	16#00300000 16#80010000	Указанный модуль занят. Выполните команду еще раз.
	16#00310000	Для указанного модуля не поддерживается подключение. Проверьте версию модуля.
	16#80000000 16#80050000 16#81010000 16#81020000 16#82020000 16#82030000 16#82060000...16#8FFF0000 16#90010000...16#FFFE0000	Произошла ошибка в сети связи. Выполните команду еще раз.
	16#80020000 16#80030000 16#81030000 16#82000000	Произошла ошибка в сети связи. Уменьшите объем передаваемых данных.
	16#80040000 16#81000000 16#82010000 16#82040000 16#82050000 16#90000000	Произошла ошибка в сети связи. Проверьте модуль и кабельные соединения. Убедитесь, что включен источник питания модуля.
16#2C01	16#00000000	Была предпринята попытка поставить в очередь более 192 команд NX_ChangeWriteMode и RestartNXUnit.
16#2C02	16#00000000	Превышено время ожидания при обмене данными.
16#2C05	---	Произошла ошибка в сети EtherCAT. Проверьте значение *UnitProxy* и конфигурацию сети EtherCAT.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
16#2C07	16#00000000	По адресу ведомого узла указанного модуля было подключено ведомое устройство, которое не может быть указано для команды. Проверьте значение *UnitProxy* и конфигурацию сети EtherCAT.

Пример программы

См. *Пример программы* на стр. 2-1119 для команды NX_WriteObj.

NX_SaveParam

Команда NX_SaveParam сохраняет данные, которые были записаны в интерфейсный модуль EtherCAT или модуль NX.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_SaveParam	Сохранение параметров модуля NX	FB	NX_SaveParam_instance NX_SaveParam Execute UnitProxy TimeOut Done Busy Error ErrorID ErrorIDEx	NX_SaveParam_instance(Execute, UnitProxy, TimeOut, Done, Busy, Error, ErrorID, ErrorIDEx);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.05 или более поздней и Sysmac Studio версии 1.06 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitProxy	Указанный модуль	Вход	Модуль, для которого нужно сохранить данные	---	---	*1
TimeOut	Время ожидания		Время ожидания Если задано значение 0, время ожидания составляет 2,0 с.	0...60 000	мс	2000 (2,0 с)

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitProxy	Подробные сведения о структуре _sNXUNIT_ID см. в разделе *Функция* на стр. 2-1004.																			
TimeOut							ОК													

Функция

Команда NX_SaveParam сохраняет данные, которые были записаны в интерфейсный модуль EtherCAT, модуль NX, подключенный к интерфейсному модулю EtherCAT, или модуль NX, подключенный к шине NX модуля ЦПУ.

Данные модуля NX сохраняются в интерфейсный модуль EtherCAT, модуль ЦПУ серии NX или в сам модуль NX.

Независимо от того, где сохраняются данные, модуль, для которого нужно сохранить данные (интерфейсный модуль EtherCAT; модуль NX, подключенный к интерфейсному модулю EtherCAT; или модуль NX, подключенный к шине NX модуля ЦПУ), указывается с помощью параметра *UnitProxy*.

После завершения сохранения данных значение *Done* меняется на ИСТИНА.
Для записи данных используйте команду *NX_WriteObj* на стр. 2-1114.
Если после выполнения этой команды будет отключено питание, значения данных с атрибутом хранения при выключенном питании сохранятся.

Параметр *TimeOut* задает время ожидания. Если ответ не возвращается в течение времени ожидания, считается, что произошел сбой связи. В этом случае данные модуля не сохраняются.

Для переменной *UnitProxy* используется структурный тип данных _sNXUNIT_ID. Значения членов структуры приведены в таблице ниже.

Имя		Значение	Описание	Тип данных
UnitProxy		Указанный модуль	Модуль, для которого нужно сохранить данные	_sNXUNIT_ID
	NodeAdr	Адрес узла	Адрес узла интерфейсного модуля	UINT
	IPAdr	IP-адрес	IP-адрес интерфейсного модуля	BYTE[5]
	UnitNo	Номер модуля	Номер модуля указанного модуля	UDINT
	Path	Путь	Информация о пути к указанному модулю	BYTE[64]
	PathLength	Допустимая длина пути *Path*	Допустимая длина пути *Path*	USINT

В *UnitProxy* следует передать переменную устройства, которая назначена указанному модулю.

Связанные команды и порядок выполнения

В зависимости от атрибутов данных, записываемых в интерфейсный модуль EtherCAT, модуль NX, подключенный к интерфейсному модулю EtherCAT, или модуль NX, подключенный к шине NX модуля ЦПУ, эту команду необходимо выполнять вместе с определенными связанными командами.

Порядок действий для каждого случая представлен ниже.

● Порядок действий 1

Для записи данных с указанными ниже атрибутами соблюдайте приведенный ниже порядок действий.

- Атрибут хранения данных при выключенном питании
- Значения обновляются при перезапуске модуля.

1 С помощью команды *NX_ChangeWriteMode* на стр. 2-998 переведите модуль в режим, допускающий запись данных.

2 Используйте команду *NX_WriteObj* на стр. 2-1114 для записи данных в модуль.

3 Используйте команду NX_SaveParam для сохранения записанных данных.

4 С помощью команды *RestartNXUnit* на стр. 2-992 перезапустите модуль.

● Порядок действий 2

Для записи данных с указанными ниже атрибутами соблюдайте приведенный ниже порядок действий.

- Атрибут хранения данных при выключенном питании
- Значения обновляются сразу же после их записи.

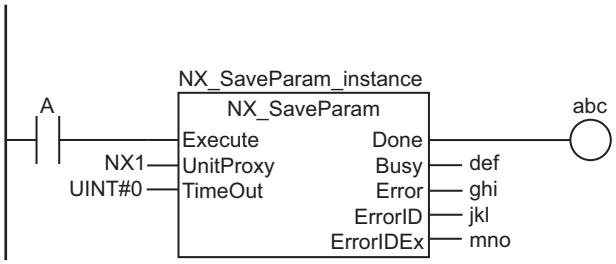
1 Используйте команду *NX_WriteObj* на стр. 2-1114 для записи данных в модуль.

2 Используйте команду NX_SaveParam для сохранения записанных данных.

Пример записи

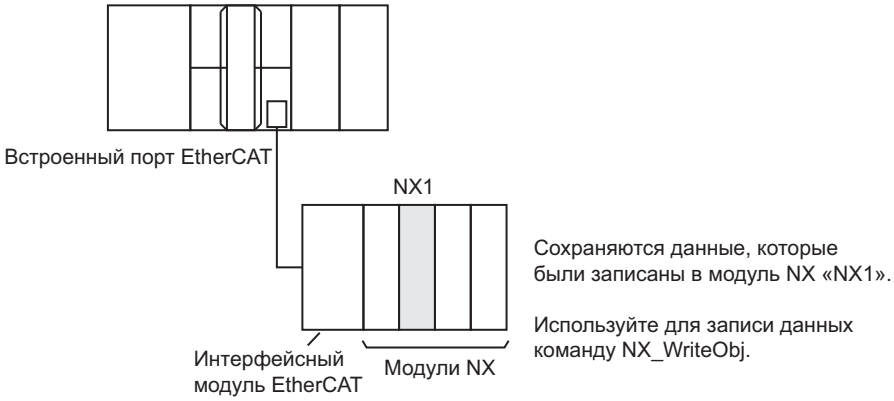
Ниже показан пример использования команды для сохранения данных, которые были записаны в модуль NX (*NX1*). Модулю NX назначена переменная с именем *NX1* и типом данных _sNXUNIT_ID.

LD



ST

```
NX_SaveParam_instance(A, NX1, UINT#0,
                      abc, def, ghi, jkl, mno);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_MBXSlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, способных участвовать в обмене сообщениями	BOOL	Эта переменная указывает возможность связи для каждого ведомого устройства. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Даже если модуль, указанный параметром *UnitProxy*, в данный момент уже сохраняет данные, эта команда не будет завершена с ошибкой. Переменная *Busy* сохранит значение ИСТИНА, а значение *Done* поменяется на ИСТИНА, когда завершится сохранение данных. Запросы на сохранение данных не ставятся в очередь.
- Даже если эта команда будет выполнена без записи данных в модуль, ошибки не произойдет.
- В некоторых модулях количество циклов записи ограничено. Дополнительные сведения см. в руководствах по соответствующим модулям.
- Для *UnitProxy* должна быть указана переменная устройства, которая назначена интерфейсному модулю EtherCAT; модулю NX, подключенному к интерфейсному модулю EtherCAT; или модулю NX, подключенному к шине NX модуля ЦПУ, на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
  Информацию о назначении переменных устройства см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.
- Чтобы записать и сохранить данные с атрибутом хранения данных при выключенном питании, выполните команду NX_SaveParam после выполнения команды *NX_WriteObj* на стр. 2-1114. Если модуль будет перезапущен до выполнения команды NX_SaveParam, будут восстановлены предыдущие данные объекта NX.
- Эта команда имеет отношение к ошибкам обмена сообщениями NX. Если одновременно будет выполнено слишком много команд, имеющих отношение к ошибкам обмена сообщениями NX, произойдет ошибка обмена сообщениями NX. Список команд, имеющих отношение к ошибкам обмена сообщениями NX, см. в разделе *A-4 Команды, связанные с ошибками обмена сообщениями NX* на стр. A-38.
- При возникновении ошибки состояние выхода *Error* поменяется на ИСТИНА. В следующей таблице приведены возможные значения переменных *ErrorID* и *ErrorIDEx* и поясняется их значение.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
16#0400	16#00000000	• Значение *UnitProxy* находится за пределами допустимого диапазона. • Значение *TimeOut* находится за пределами допустимого диапазона.
16#0419	16#00000000	Неправильный тип данных *UnitProxy*.
16#2C00	16#00000401	Указанный модуль не поддерживает команду.
	16#00001001 16#00001002 16#00170000 16#00200000 16#00210000	Неправильный входной, выходной или входной-выходной параметр. Проследите, чтобы для входного, выходного или входного-выходного параметра использовался предусмотренный параметр.
	16#00001010	Размер данных указанного объекта NX не согласуется с размером данных, который указан в параметре *WriteDat*.
	16#00001101	Не указан правильный модуль. Проверьте модуль.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
	16#0000110B	Размер прочитанных данных слишком велик. Убедитесь, что параметры читаемых данных верны и что читаемые данные указаны верно.
	16#00001110	Не существует объекта, соответствующего значению *Obj.Index*.
	16#00001111	Не существует объекта, соответствующего значению *Obj.Subindex*.
	16#00002101	Указанный объект NX не может быть записан.
	16#00002110	Значение *WriteDat* выходит за диапазон значений, которые могут быть записаны для объекта NX.
	16#00002210	Указанный модуль не находится в режиме, в котором разрешена запись данных.
	16#00002213	Выполнение команды было невозможно, поскольку указанный модуль выполнял проверку ввода-вывода. Выполните команду после завершения проверки ввода-вывода.
	16#00002230	Состояние указанного модуля не согласуется со значением исходного объекта NX для чтения или целевого объекта NX для записи. Если значение *Obj.Index* находится в диапазоне от 0x6000 до 0x6FFF или от 0x7000 до 0x7FFF, выполните указанные ниже действия. • Удалите исходный объект NX для чтения или целевой объект NX для записи из параметров распределения данных ввода-вывода. • Сбросьте ошибку для указанного модуля. • Переведите указанный модуль в режим, в котором не разрешена запись данных.
	16#00002231	Выполнение команды было невозможно, поскольку указанный модуль выполнял инициализацию. Дождитесь, пока модуль начнет работать в обычном режиме, а затем выполните команду.
	16#0000250F	Не удалось получить доступ к оборудованию. Выполните команду еще раз.
	16#00002601 16#00002602 16#00100000	Указанный модуль не поддерживает эту команду. Проверьте версию модуля.
	16#00002603	Не удалось выполнить команду. Выполните команду еще раз. Убедитесь, что в параметрах выбора используемых каналов активирован (**Enabled**) хотя бы один канал.
	16#00002621	Модуль NX не находится в состоянии, в котором он может подтвердить команду. Подождите некоторое время, а затем выполните команду еще раз.
	16#00010000	Указанного модуля не существует. Убедитесь, что конфигурация модуля верна.
	16#00110000	Указанного номера порта не существует. Убедитесь, что конфигурация модуля верна.
	16#00120000 16#00130000 16#00150000 16#00160000	Неверное значение *UnitProxy*. Снова задайте переменную, которая указывает указанный интерфейсный модуль EtherCAT.
	16#00140000	Указан неправильный адрес узла. Убедитесь, что конфигурация модуля верна.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
	16#00300000 16#80010000	Указанный модуль занят. Выполните команду еще раз.
	16#00310000	Для указанного модуля не поддерживается подключение. Проверьте версию модуля.
	16#80000000 16#80050000 16#81010000 16#81020000 16#82020000 16#82030000 16#82060000...16#8FFF0000 16#90010000...16#FFFE0000	Произошла ошибка в сети связи. Выполните команду еще раз.
	16#80020000 16#80030000 16#81030000 16#82000000	Произошла ошибка в сети связи. Уменьшите объем передаваемых данных.
	16#80040000 16#81000000 16#82010000 16#82040000 16#82050000 16#90000000	Произошла ошибка в сети связи. Проверьте модуль и кабельные соединения. Убедитесь, что включен источник питания модуля.
16#2C01	16#00000000	Превышено количество команд, которые могут выполняться одновременно.
16#2C02	16#00000000	Превышено время ожидания при обмене данными.

Пример программы

См. *Пример программы* на стр. 2-1119 для команды NX_WriteObj.

PLC_ReadTotalPowerOnTime

Команда PLC_ReadTotalPowerOnTime считывает общее время работы при включенном питании из указанного модуля ЦПУ.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
PLC_ReadTotalPowerOnTime	Чтение общего времени работы ПЛК	FUN	PLC_ReadTotalPowerOnTime EN — ENO UnitType — Out	Out:=PLC_ReadTotalPowerOnTime(UnitType);

Меры предосторожности для обеспечения надлежащей эксплуатации

Эту команду можно использовать для модулей ЦПУ NX102 и NX1P2.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitType	Тип модуля	Вход	Указывает целевой модуль	_CPU_UNIT	---	_CPU_UNIT
Out	Общее время работы при включенном питании	Выход	Выдает прочитанное значение общего времени работы при включенном питании.	Зависит от типа данных.		---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitType	Сведения о перечислителях перечислимого типа _ePLC_UNIT_TYPE см. в разделе *Функция* на стр. 2-1010.																			
Out																ОК				

Функция

Команда PLC_ReadTotalPowerOnTime считывает приблизительное значение общего времени работы при включенном питании из указанного модуля ЦПУ.
Погрешность составляет 1 час на месяц.

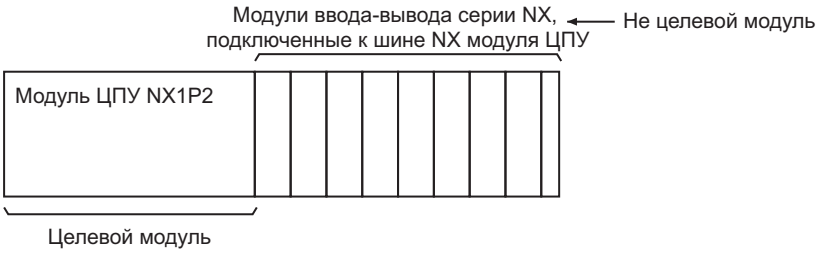
Необходимо указать параметр *UnitType* для модуля, из которого должны быть прочитаны данные.

Для параметра *UnitType* используется перечислимый тип данных _ePLC_UNIT_TYPE. Значение перечислителя приведено в таблице ниже.

Перечислитель	Значение
_CPU_UNIT	Указывается модуль ЦПУ.

Целевой модуль

● При использовании модуля ЦПУ NX1P2



Целевым модулем для чтения данных может быть только модуль ЦПУ NX1P2.
Для указания целевого модуля задайте параметр *UnitType* с перечислителем _CPU_UNIT перечислимого типа _ePLC_UNIT_TYPE.
Модули ввода-вывода серии NX, подключенные к шине NX модуля ЦПУ, не могут выступать в качестве целевого модуля для чтения.

Дополнительная информация

Чтобы прочитать общее время работы при включенном питании для модуля ввода-вывода серии NX, используйте команду *NX_ReadTotalPowerOnTime* на стр. 2-1013.

Дополнительная информация

При выполнении этой команды в средстве моделирования (Simulator) значение *Out* всегда равно T#0s.

Меры предосторожности для обеспечения надлежащей эксплуатации

Если указанный модуль не является целевым модулем для чтения, произойдет ошибка.
Выход *ENO* будет содержать ЛОЖЬ, а значение *Out* не изменится.

Пример программы

В данном примере в программе создаются две переменные: Maintenance_Mode и Run_Mode.
Если в режиме Maintenance_Mode будет нажата кнопка чтения общего времени работы, команда произведет чтение общего времени работы при включенном питании из модуля ЦПУ NX1P2.
Если общее время работы при включенном питании превышает 5 лет, загорается лампа, предупреждающая о необходимости замены модуля.
После того как модуль будет заменен и будет нажата кнопка завершения замены модуля, предупреждающая лампа погаснет.

Система имеет следующую конфигурацию:

Модуль	Описание
Модуль 1	Модуль 1, который подключен к шине NX модуля ЦПУ серии NX. Модуль ввода-вывода серии NX (ID)
Модуль 2	Модуль 2, который подключен к шине NX модуля ЦПУ серии NX. Модуль ввода-вывода серии NX (IO)
Модуль 3	Модуль ЦПУ NX1P2 (целевой модуль для чтения общего времени работы при включенном питании)

Программа на языке ST

Внешние переменные	Переменная	Тип данных	Комментарий
	J01_Ch1_In00	BOOL	Кнопка режима обслуживания
	J01_Ch1_In01	BOOL	Кнопка для чтения общего времени работы при включенном питании
	J02_Ch1_Out00	BOOL	Лампа предупреждения о необходимости замены модуля
	Maintenance_Mode	BOOL	
	Run_Mode	BOOL	
	PushButton_Read	BOOL	
	Lamp_Warning_UnitLifeTime	BOOL	
	PowerOnTime	TIME	
	R_TRIG_instance1	R_TRIG	
	PushButton_Read_R_TRIG	BOOL	
	RS_instance	RS	

```
// Получение состояния кнопки.
Maintenance_Mode := J01_Ch1_In00;
Run_Mode := NOT(J01_Ch1_In00);
PushButton_Read := J01_Ch1_In01;

R_TRIG_instance1(clk:=PushButton_Read, Q=>PushButton_Read_R_TRIG);

// Чтение общего времени работы.
PowerOnTime := PLC_ReadTotalPowerOnTime(EN:=(Maintenance_Mode & PushButton_Read_R_T
RIG),
  UnitType:=_CPU_UNIT);

RS_instance( Set:=(PowerOnTime > T#1825d),
  Reset1:=Maintenance_Mode,
  Q1=>Lamp_Warning_UnitLifeTime);

// Включение сигнальной лампы.
J02_Ch1_Out00 := Lamp_Warning_UnitLifeTime;
```

NX_ReadTotalPowerOnTime

Команда NX_ReadTotalPowerOnTime считывает общее время работы при включенном питании из интерфейсного модуля или модуля NX.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_ReadTotal PowerOnTime	Чтение общего времени работы модуля NX	FB	NX_ReadTotalPowerOnTime_instance NX_ReadTotalPowerOnTime Execute — Done UnitProxy — Busy Error ErrorID ErrorIDEx TotalPowerOnTime	NX_ReadTotalPowerONTime_instance(Execute, UnitProxy, Done, Busy, Error, ErrorID, ErrorIDEx, TotalPowerOnTime);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitProxy	Указанный модуль	Вход	Указывает целевой модуль NX	---	---	*1
TotalPower OnTime	Общее время работы	Выход	Хранит прочитанное значение общего времени работы при включенном питании.	Зависит от типа данных.		0

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitProxy	Подробные сведения о структуре _sNXUNIT_ID см. в разделе *Функция* на стр. 2-1013.																			
TotalPower OnTime																OK				

Функция

Команда NX_ReadTotalPowerOnTime считывает приблизительное значение общего времени работы при включенном питании из интерфейсного модуля; модуля NX, подключенного к интерфейсному модулю; или модуля NX, подключенного к шине NX модуля ЦПУ.
Погрешность составляет 1 час на месяц.

Модуль, из которого должно быть прочитано общее время работы, указывается с помощью параметра *UnitProxy*.

После завершения чтения общего времени работы значение переменной *Done* меняется на ИСТИНА.

Для переменной *UnitProxy* используется структурный тип данных _sNXUNIT_ID. Значения членов структуры приведены в таблице ниже.

Имя		Значение	Описание	Тип данных
UnitProxy		Указанный модуль	Указанный модуль	_sNXUNIT_ID
	NodeAdr	Адрес узла	Адрес узла интерфейсного модуля	UINT
	IPAdr	IP-адрес	IP-адрес интерфейсного модуля	BYTE[5]
	UnitNo	Номер модуля	Номер модуля указанного модуля NX	UDINT
	Path	Путь	Информация о пути к указанному модулю	BYTE[64]
	PathLength	Допустимая длина пути *Path*	Допустимая длина пути *Path*	USINT

В *UnitProxy* необходимо передать переменную устройства, назначенную указанному интерфейсному модулю или модулю NX, подключенному к интерфейсному модулю, либо модулю NX, подключенному к шине NX модуля ЦПУ.

Комбинации версий

Ниже перечислены комбинации версий интерфейсного модуля, подключенного к модулю ЦПУ; модуля NX, подключенного к интерфейсному модулю; и модуля NX, подключенного к шине NX модуля ЦПУ, при которых возможно чтение общего времени работы при включенном питании.

● Ведомый терминал EtherCAT

Модуль	Версия модуля NX	Версия интерфейсного модуля EtherCAT
Модуль дискретных входов-выходов	Версия 1.0 или более поздняя	Версия 1.2 или более поздняя
Модуль аналоговых входов-выходов		
Конструктивный модуль		
Модуль интерфейса позиционирования	Версия 1.1 или более поздняя	
Модуль температурных входов		

● Модуль NX в стойке модуля ЦПУ NX102 или NX1P2

Модуль	Версия модуля NX
Модуль дискретных входов-выходов	Версия 1.0 или более поздняя
Модуль аналоговых входов-выходов	
Конструктивный модуль	
Модуль интерфейса позиционирования	Версия 1.1 или более поздняя
Модуль температурных входов	

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_MBXSlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, способных участвовать в обмене сообщениями	BOOL	Эта переменная указывает возможность связи для каждого ведомого устройства. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_NXB_UnitMsgActiveTbl[i]	Состояние активации обмена сообщениями модуля NX	BOOL	В этой таблице указываются ведомые устройства, которые способны участвовать в обмене сообщениями. С помощью этой переменной можно проверить, возможен ли обмен сообщениями с конкретным ведомым устройством.

Дополнительная информация

При выполнении этой команды в средстве моделирования (Simulator) выход *Busy* переходит в состояние ИСТИНА только на один цикл выполнения задачи после перехода *Execute* из ЛОЖЬ в ИСТИНА.
В следующем цикле выполнения задачи выход *Busy* возвращается в состояние ЛОЖЬ, а выход *Done* переходит в состояние ИСТИНА.
Прочитанное значение *TotalPowerOnTime* будет равно *0*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Для *UnitProxy* должна быть указана переменная устройства, которая назначена интерфейсному модулю EtherCAT; модулю NX, подключенному к интерфейсному модулю EtherCAT; или модулю NX, подключенному к шине NX модуля ЦПУ, на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
  Информацию о назначении переменных устройства см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.
  Если в параметре *UnitProxy* будет указан модуль ЦПУ серии NX, произойдет ошибка.
- Имеются ограничения на количество модулей, которые зависят от интерфейсного модуля. Дополнительные сведения см. в руководстве по используемому интерфейсному модулю.
- При возникновении ошибки состояние выхода *Error* поменяется на ИСТИНА. В следующей таблице приведены возможные значения переменных *ErrorID* и *ErrorIDEx* и поясняется их значение.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
16#0400	16#00000000	Значение *UnitProxy* находится за пределами допустимого диапазона.
16#0419	16#00000000	Неправильный тип данных *UnitProxy*.
16#2C00	16#00000401	Указанный модуль не поддерживает команду.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
	16#00001001 16#00001002 16#00170000 16#00200000 16#00210000	Неправильный входной, выходной или входной-выходной параметр. Проследите, чтобы для входного, выходного или входного-выходного параметра использовался предусмотренный параметр.
	16#00001010	Размер данных указанного объекта NX не согласуется с размером данных, который указан в параметре *WriteDat*.
	16#00001101	Неверный модуль. Проверьте модуль.
	16#0000110B	Размер прочитанных данных слишком велик. Убедитесь, что параметры читаемых данных верны и что читаемые данные указаны верно.
	16#00001110	Не существует объекта, соответствующего значению *Obj.Index*.
	16#00001111	Не существует объекта, соответствующего значению *Obj.Subindex*.
	16#00002101	Указанный объект NX не может быть записан.
	16#00002110	Значение *WriteDat* выходит за диапазон значений, которые могут быть записаны для объекта NX.
	16#00002210	Указанный модуль не находится в режиме, в котором разрешена запись данных.
	16#00002213	Выполнение команды было невозможно, поскольку указанный модуль выполнял проверку ввода-вывода. Выполните команду после завершения проверки ввода-вывода.
	16#00002230	Состояние указанного модуля не согласуется со значением исходного объекта NX для чтения или целевого объекта NX для записи. Если значение *Obj.Index* находится в диапазоне от 0x6000 до 0x6FFF или от 0x7000 до 0x7FFF, выполните указанные ниже действия. • Удалите исходный объект NX для чтения или целевой объект NX для записи из параметров распределения данных ввода-вывода. • Сбросьте ошибку для указанного модуля. • Переведите указанный модуль в режим, в котором не разрешена запись данных.
	16#00002231	Выполнение команды было невозможно, поскольку указанный модуль выполнял инициализацию. Дождитесь, пока модуль начнет работать в обычном режиме, а затем выполните команду.
	16#0000250F	Не удалось получить доступ к оборудованию. Выполните команду еще раз.
	16#00002601 16#00002602 16#00100000	Указанный модуль не поддерживает эту команду. Проверьте версию модуля.
	16#00002603	Не удалось выполнить команду. Выполните команду еще раз. Убедитесь, что в параметрах выбора используемых каналов активирован (**Enabled**) хотя бы один канал.
	16#00002621	Модуль NX не находится в состоянии, в котором он может подтвердить команду. Подождите некоторое время, а затем выполните команду еще раз.
	16#00010000	Указанного модуля не существует. Убедитесь, что конфигурация модуля верна.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
	16#00110000	Указанного номера порта не существует. Убедитесь, что конфигурация модуля верна.
	16#00120000 16#00130000 16#00150000 16#00160000	Неверное значение *UnitProxy*. Снова задайте переменную, которая указывает указанный интерфейсный модуль EtherCAT.
	16#00140000	Указан неправильный адрес узла. Убедитесь, что конфигурация модуля верна.
	16#00300000 16#80010000	Указанный модуль занят. Выполните команду еще раз.
	16#00310000	Для указанного модуля не поддерживается подключение. Проверьте версию модуля.
	16#80000000 16#80050000 16#81010000 16#81020000 16#82020000 16#82030000 16#82060000...16#8FFF0000 16#90010000...16#FFFE0000	Произошла ошибка в сети связи. Выполните команду еще раз.
	16#80020000 16#80030000 16#81030000 16#82000000	Произошла ошибка в сети связи. Уменьшите объем передаваемых данных.
	16#80040000 16#81000000 16#82010000 16#82040000 16#82050000 16#90000000	Произошла ошибка в сети связи. Проверьте модуль и кабельные соединения. Убедитесь, что включен источник питания модуля.
16#2C01	16#00000000	Превышено количество команд, которые могут выполняться одновременно.
16#2C02	16#00000000	Превышено время ожидания при обмене данными.
16#2C08	16#00000000	Не удалось прочитать общее время наработки.

Пример программы

В данном примере программы создаются два режима: режим обслуживания и режим работы.
Если в режиме обслуживания будет нажата кнопка чтения общего времени работы, будет произведено чтение общего времени работы при включенном питании для модуля 3 (модуль задан заранее).
Если общее время работы при включенном питании превышает 5 лет, загорается лампа, предупреждающая о необходимости замены модуля.
Если после замены модуля будет нажата кнопка завершения замены модуля, предупреждающая лампа погаснет.

Система имеет следующую конфигурацию.

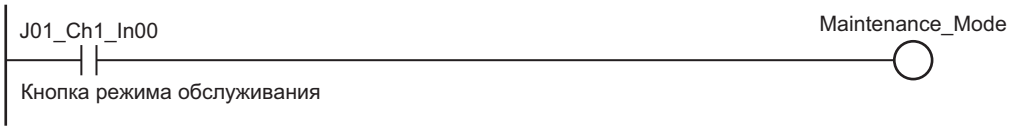
Модуль	Описание
Модуль 1	Модуль NX (ID)
Модуль 2	Модуль NX (OD)
Модуль 3	Модуль NX (модуль, из которого должно быть прочитано общее время работы)

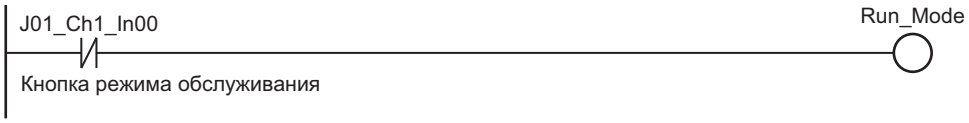
Программа на языке LD

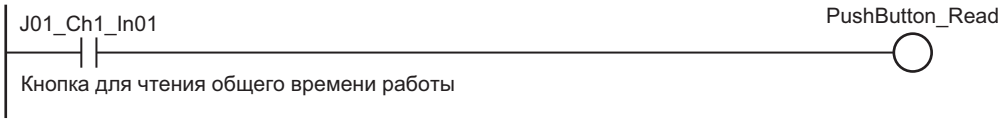
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Maintenance_Mode	BOOL	ЛОЖЬ	Режим обслуживания
	Run_Mode	BOOL	ЛОЖЬ	Режим «Работа»
	PushButton_Read	BOOL	ЛОЖЬ	Чтение общего времени работы
	PushButton_Changed	BOOL	ЛОЖЬ	Завершение замены модуля
	Lamp_Warning_UnitLifeTime	BOOL	ЛОЖЬ	Предупреждение о необходимости замены модуля
	Read	BOOL	ЛОЖЬ	
	instance	NX_ReadTotalPowerOnTime		

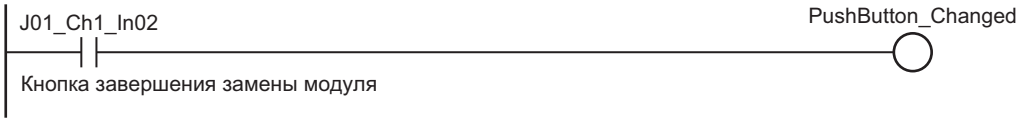
Внешние переменные	Переменная	Тип данных	Комментарий
	NX_Unit	_sNXUNIT_ID	
	J01_Ch1_In00	BOOL	Кнопка режима обслуживания
	J01_Ch1_In01	BOOL	Кнопка для чтения общего времени работы при включенном питании
	J01_Ch1_In02	BOOL	Кнопка завершения замены модуля
	J02_Ch1_Out00	BOOL	Лампа предупреждения о необходимости замены модуля

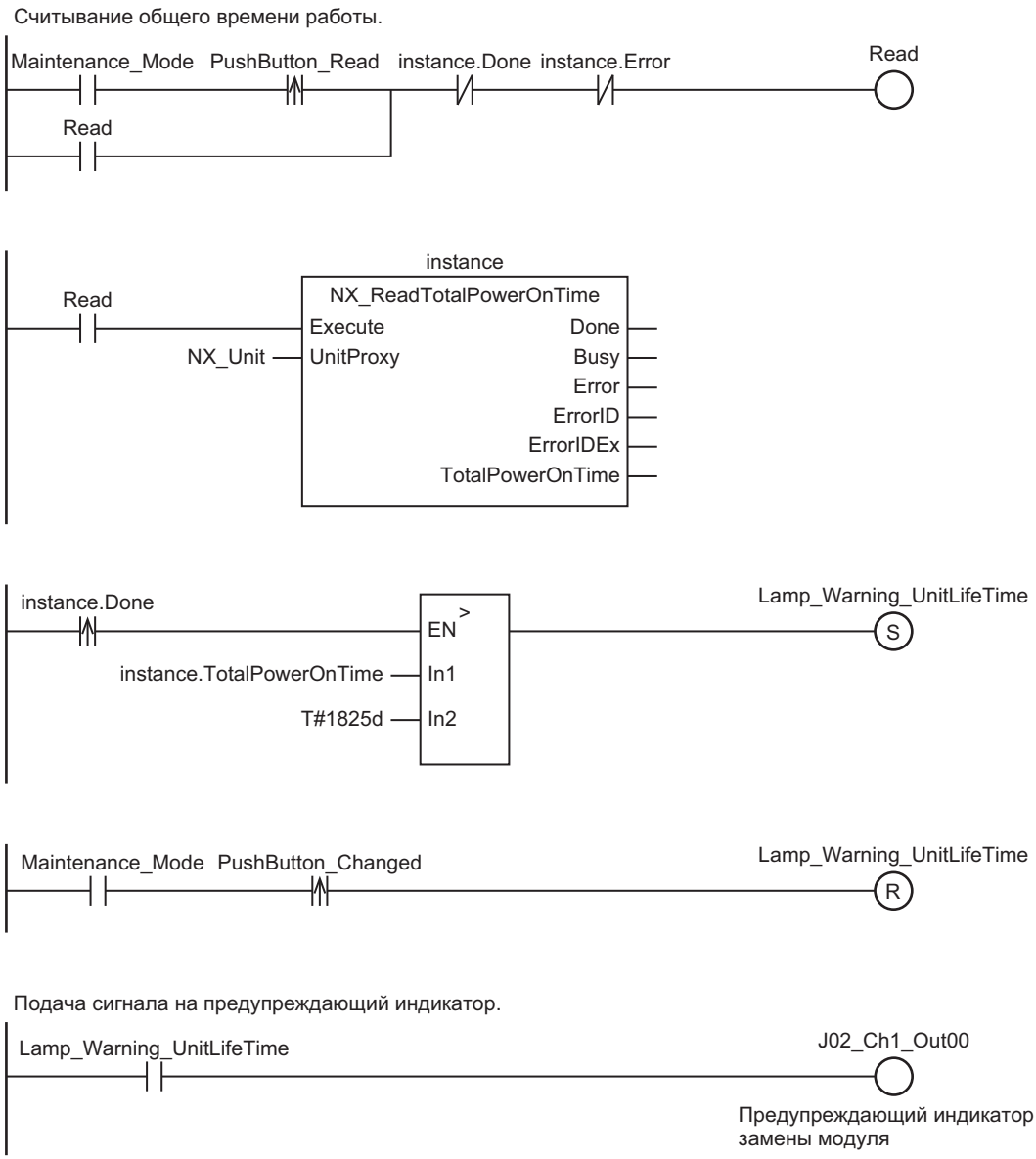
Получение состояния кнопки.











Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Maintenance_Mode	BOOL	ЛОЖЬ	Режим обслуживания
	Run_Mode	BOOL	ЛОЖЬ	Режим «Работа»
	PushButton_Read	BOOL	ЛОЖЬ	Чтение общего времени работы
	PushButton_Changed	BOOL	ЛОЖЬ	Завершение замены модуля
	Lamp_Warning_UnitLifeTime	BOOL	ЛОЖЬ	Предупреждение о необходимости замены модуля
	Read	BOOL	ЛОЖЬ	
	instance	NX_ReadTotalPowerOnTime		

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	RS_instance	RS		
	RS_instance2	RS		
	R_TRIG_instance1	R_TRIG		
	R_TRIG_instance2	R_TRIG		
	R_TRIG_instance3	R_TRIG		
	PushButton_Read_R_TRIG	BOOL		
	instance_Done_R_TRIG	BOOL		
	PushButton_Change_R_TRIG	BOOL		

Внешние переменные	Переменная	Тип данных	Комментарий
	NX_Unit	_sNXUNIT_ID	
	J01_Ch1_In00	BOOL	Кнопка режима обслуживания
	J01_Ch1_In01	BOOL	Кнопка для чтения общего времени работы при включенном питании
	J01_Ch1_In02	BOOL	Кнопка завершения замены модуля
	J02_Ch1_Out00	BOOL	Лампа предупреждения о необходимости замены модуля

```
// Получение состояния кнопки.
Maintenance_Mode := J01_Ch1_In00;
Run_Mode := NOT(J01_Ch1_In00);
PushButton_Read := J01_Ch1_In01;
PushButton_Changed := J01_Ch1_In02;

R_TRIG_instance1(Clk:= PushButton_Read, Q=>PushButton_Read_R_TRIG);

// Чтение общего времени работы при включенном питании.
Rs_instance( Set:= (Maintenance_Mode & PushButton_Read_R_TRIG),
  Reset1:=((instance.Done) OR (instance.Error)),
  Q1=>Read);
instance(Execute:=Read, UnitProxy:=NX_Unit);

R_TRIG_instance2(Clk:= instance.Done, Q=>instance_Done_R_TRIG);
R_TRIG_instance3(Clk:= PushButton_Changed, Q=>PushButton_Changed_R_TRIG);

RS_instance2(Set:=(instance_Done_R_TRIG & (instance.TotalPowerOnTime>T#1825d)),
  Reset1:=(Maintenance_Mode & PushButton_Changed_R_TRIG),
  Q1=>Lamp_Warning_UnitLifeTime);

// Включение сигнальной лампы.
J02_Ch1_Out00 := Lamp_Warning_UnitLifeTime;
```

Команды для управления программами

Команда	Имя	Стр.
PrgStart	Активация программы	стр. 2-1022
PrgStop	Деактивация программы	стр. 2-1032
PrgStatus	Чтение статуса программы	стр. 2-1052

PrgStart

Команда PrgStart разрешает выполнение указанной программы.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
PrgStart	Активация программы	FUN	(@)PrgStart EN PrgName isFirstRun Out	Out:=PrgStart(PrgName, isFirstRun);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
PrgName	Имя программы	Вход	Имя указанной программы	Макс. 128 байт (127 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	*1
isFirstRun	Активация флага первого цикла программы		Работа системной переменной *P_First_Run* в первом цикле выполнения задачи, в котором выполняется программа ИСТИНА: переход в состояние «ИСТИНА». ЛОЖЬ: переход в состояние «ЛОЖЬ».	Зависит от типа данных.		ИСТИНА
Out	Флаг нормального завершения	Выход	Флаг нормального завершения ИСТИНА: нормальное завершение ЛОЖЬ: завершение с ошибкой	Зависит от типа данных.	---	---

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
PrgName																				OK
isFirstRun	OK																			
Out	OK																			

Функция

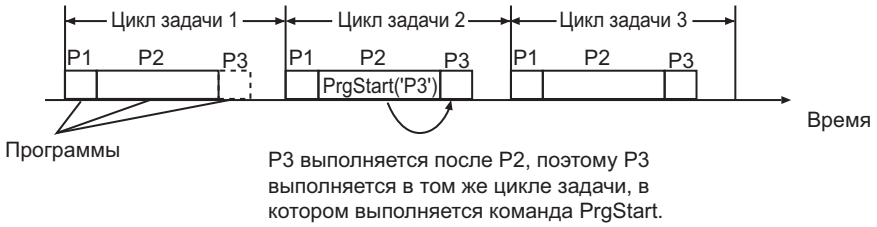
Команда PrgStart разрешает выполнение программы, указанной параметром *PrgName*.
Указанная программа выполняется в следующий раз, когда наступает время ее выполнения. Даже если выполнение указанной программы уже разрешено, ошибки не возникает.
Указанная программа может находиться в той же задаче, что и эта команда, или в другой задаче.
Значение переменной *Out* равно ИСТИНА при нормальном завершении команды и ЛОЖЬ — при завершении команды с ошибкой.

Пример работы команды для случая, когда указана программа в текущей задаче

Ниже рассматривается пример работы команды для случая, когда в команде указана программа, которая находится в той же задаче, что и сама команда.

● Активация программы, выполняемой после команды PrgStart

- В данном примере в одной задаче имеются три программы: P1, P2 и P3.
- Программа P3 деактивируется, начиная с цикла выполнения задачи 1.
- В программе P2 цикла выполнения задачи 2 выполняется команда PrgStart, в которой указана программа P3.
- Программа P3 выполняется после программы P2, поэтому программа P3 выполняется в цикле выполнения задачи 2.
- После этого программа P3 остается активированной, даже если команда PrgStart с указанной программой P3 больше не выполняется.

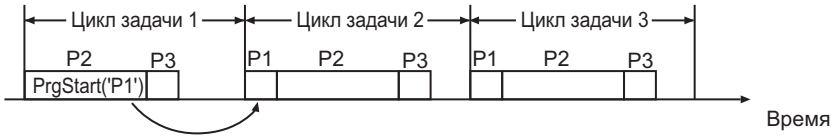


P3 выполняется после P2, поэтому P3 выполняется в том же цикле задачи, в котором выполняется команда PrgStart.

● Активация программы, выполняемой до команды PrgStart

- В данном примере в одной задаче имеются три программы: P1, P2 и P3.
- Программа P1 деактивируется, начиная с цикла выполнения задачи 1.

- В программе P2 цикла выполнения задачи 1 выполняется команда PrgStart, в которой указана программа P1.
- Программа P1 выполняется до программы P2, поэтому программа P1 выполняется в цикле выполнения задачи 2.
- После этого программа P1 остается активированной, даже если команда PrgStart с указанной программой P1 больше не выполняется.



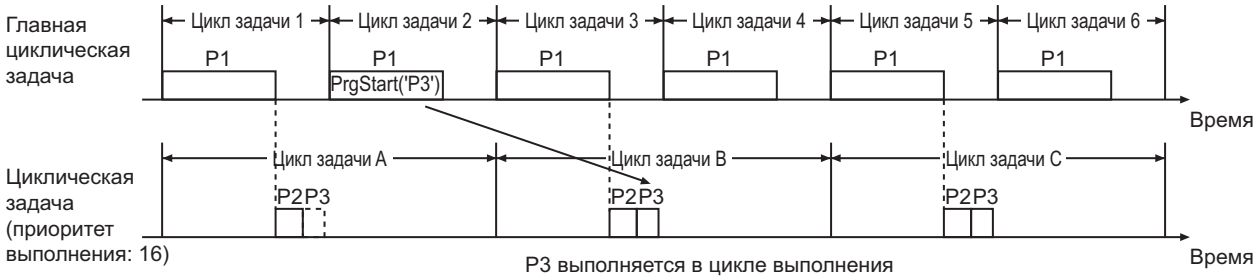
P1 выполняется перед P2, поэтому P1 выполняется в следующем цикле, т. е. после цикла, в котором выполняется команда PrgStart.

Пример работы команды для случая, когда указана программа в другой задаче

Ниже рассматривается пример работы команды для случая, когда в команде указана программа, которая находится не в той же задаче, что и команда, а в другой задаче.

● Активация программы в задаче с более низким приоритетом, чем у текущей задачи

- В данном примере имеются три программы. Программа P1 находится в главной циклической задаче, а программы P2 и P3 находятся в циклической задаче.
- Программа P3 деактивируется, начиная с цикла выполнения задачи A циклической задачи.
- В программе P1 цикла выполнения задачи 2 главной циклической задачи выполняется команда PrgStart, в которой указана программа P3.
- Программа P3 выполняется в цикле выполнения задачи B циклической задачи, которая выполняется после выполнения команды PrgStart.
- После этого программа P3 остается активированной, даже если команда PrgStart с указанной программой P3 больше не выполняется.

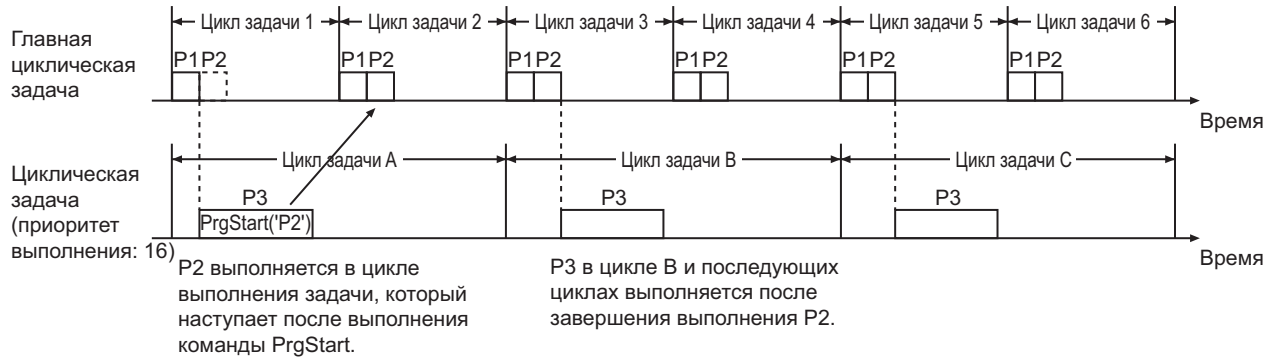


P3 выполняется в цикле выполнения задачи, который наступает после выполнения команды PrgStart.

● Активация программы в задаче с более высоким приоритетом, чем у текущей задачи

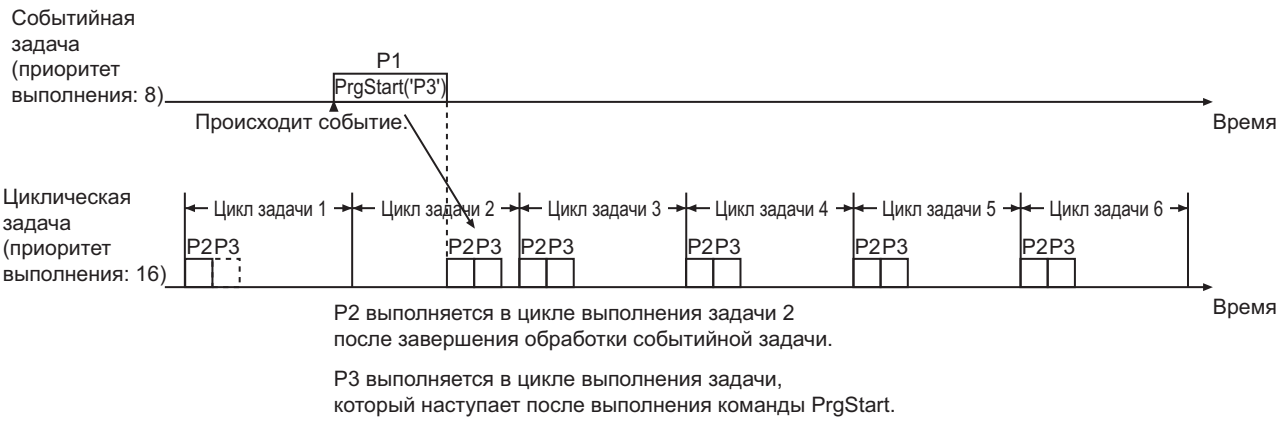
- В данном примере имеются три программы. Программы P1 и P2 находятся в главной циклической задаче, а программа P3 находится в циклической задаче.

- Программа P2 деактивируется, начиная с цикла выполнения задачи 1 главной циклической задачи.
- В программе P3 цикла выполнения задачи A циклической задачи выполняется команда PrgStart, в которой указана программа P2.
- Программа P2 выполняется в цикле выполнения задачи 2 главной циклической задачи, которая выполняется после выполнения команды PrgStart.
- После этого программа P2 остается активированной, даже если команда PrgStart с указанной программой P2 больше не выполняется.
- Главная циклическая задача обладает более высоким приоритетом выполнения, чем циклическая задача, поэтому программа P3 в цикле выполнения задачи B и последующих циклах выполняется после завершения обработки программы P2.



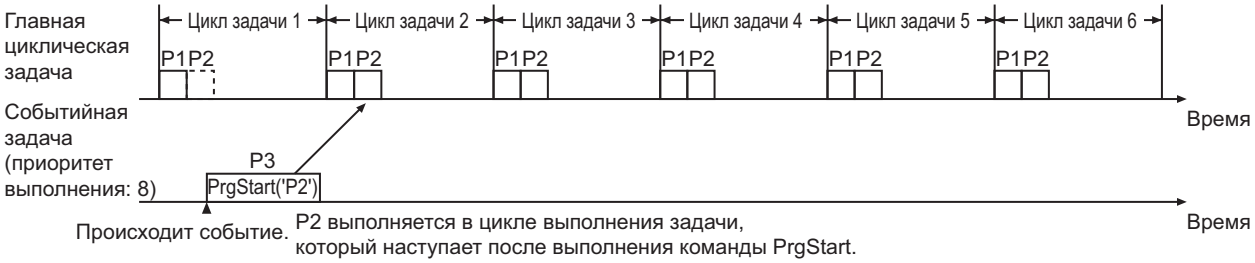
● Активация программы в задаче с более низким приоритетом из событийной задачи

- В данном примере имеются три программы. Программа P1 находится в событийной задаче (приоритет выполнения: 8), а программы P2 и P3 находятся в циклической задаче (приоритет выполнения: 16).
- Программа P3 деактивируется, начиная с цикла выполнения задачи 1 циклической задачи.
- В событийной задаче выполняется команда PrgStart, в которой указана программа P3.
- Когда выполняется событийная задача, в цикле выполнения задачи 2 циклической задачи после завершения обработки событийной задачи выполняются программы P2 и P3.
- В результате программа P3 выполняется уже в цикле выполнения задачи 2 циклической задачи, так как он следует после выполнения команды PrgStart.
- После этого программа P3 остается активированной, даже если команда PrgStart с указанной программой P3 больше не выполняется.



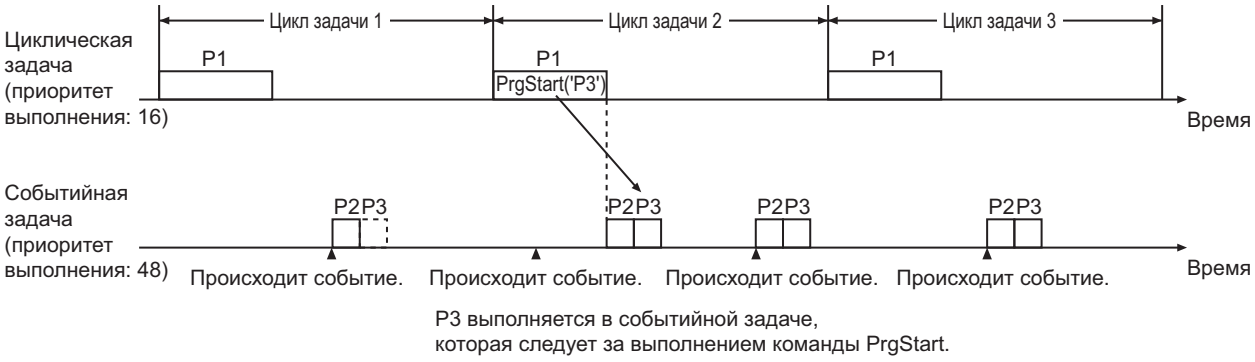
● Активация программы в задаче с более высоким приоритетом из событийной задачи

- В данном примере имеются три программы. Программы P1 и P2 находятся в главной циклической задаче, а программа P3 находится в событийной задаче.
- Программа P2 деактивируется, начиная с цикла выполнения задачи 1 главной циклической задачи.
- В событийной задаче выполняется команда PrgStart, в которой указана программа P2.
- Программа P2 выполняется в цикле выполнения задачи 2 главной циклической задачи, которая выполняется после выполнения команды PrgStart.
- После этого программа P2 остается активированной, даже если команда PrgStart с указанной программой P2 больше не выполняется.



● Активация программы в событийной задаче с более низким приоритетом из циклической задачи

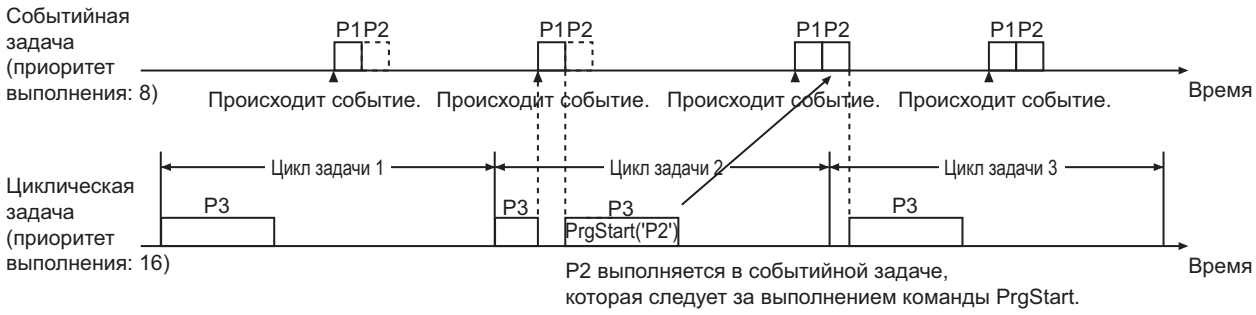
- В данном примере имеются три программы. Программа P1 находится в циклической задаче (приоритет выполнения: 16), а программы P2 и P3 находятся в событийной задаче (приоритет выполнения: 48).
- Программа P3 в событийной задаче деактивирована.
- В циклической задаче выполняется команда PrgStart, в которой указана программа P3.
- Программа P3 выполняется в событийной задаче, которая выполняется после выполнения команды PrgStart.
- После этого программа P3 остается активированной, даже если команда PrgStart с указанной программой P3 больше не выполняется.



● Активация программы в событийной задаче с более высоким приоритетом из циклической задачи

- В данном примере имеются три программы. Программы P1 и P2 находятся в событийной задаче (приоритет выполнения: 8), а программа P3 находится в циклической задаче (приоритет выполнения: 16).

- Программа P2 в событийной задаче деактивирована.
- В циклической задаче выполняется команда PrgStart, в которой указана программа P2.
- Программа P2 выполняется в событийной задаче, которая выполняется после выполнения команды PrgStart.
- После этого программа P2 остается активированной, даже если команда PrgStart с указанной программой P2 больше не выполняется.



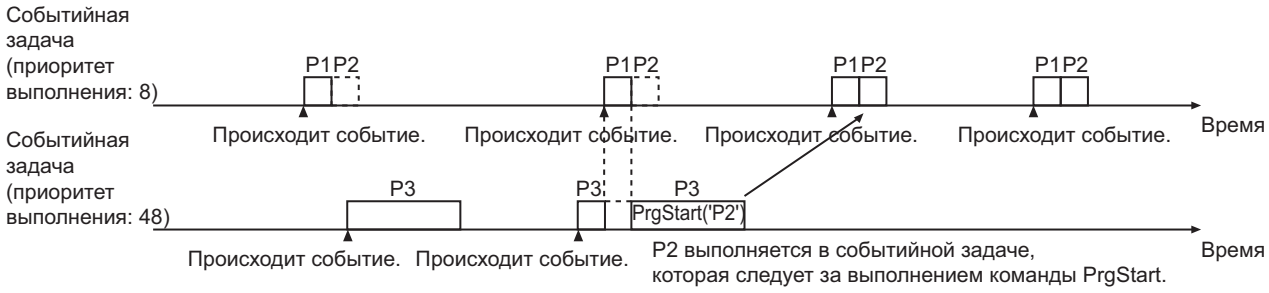
● Активация программы в событийной задаче с более низким приоритетом из другой событийной задачи

- В данном примере имеются три программы. Программа P1 находится в событийной задаче (приоритет выполнения: 8), а программы P2 и P3 находятся в событийной задаче (приоритет выполнения: 48).
- Программа P3 в событийной задаче (приоритет выполнения: 48) деактивирована.
- В событийной задаче (приоритет выполнения: 8) выполняется команда PrgStart, в которой указана программа P3.
- Программа P3 выполняется в событийной задаче (приоритет выполнения: 48), которая выполняется после выполнения команды PrgStart.
- После этого программа P3 остается активированной, даже если команда PrgStart с указанной программой P3 больше не выполняется.

● Активация программы в событийной задаче с более высоким приоритетом из другой событийной задачи

- В данном примере имеются три программы. Программы P1 и P2 находятся в событийной задаче (приоритет выполнения: 8), а программа P3 находится в событийной задаче (приоритет выполнения: 48).
- Программа P2 в событийной задаче (приоритет выполнения: 8) деактивирована.
- В событийной задаче (приоритет выполнения: 48) выполняется команда PrgStart, в которой указана программа P2.
- Программа P2 выполняется в событийной задаче (приоритет выполнения: 8), которая выполняется после выполнения команды PrgStart.

- После этого программа P2 остается активированной, даже если команда PrgStart с указанной программой P2 больше не выполняется.



Активация флага первого цикла программы (*isFirstRun*)

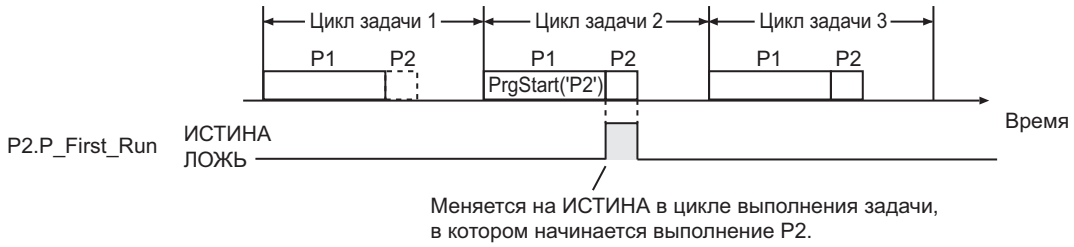
Параметр *isFirstRun* определяет, разрешена ли работа системной переменной *P_First_Run* в соответствии с приведенной ниже таблицей.
Если *isFirstRun* = ИСТИНА при выполнении команды, флаг *P_First_Run* переходит в состояние ИСТИНА на один цикл выполнения задачи, когда начинается выполнение программы.
Если *isFirstRun* = ЛОЖЬ при выполнении команды, флаг *P_First_Run* остается в состоянии ЛОЖЬ, даже если начинается выполнение программы.
Используйте параметр *isFirstRun*, когда нужно выполнять определенную обработку в начале выполнения программы, только если выполняются определенные условия.
Если заданные условия соблюдаются, значение *isFirstRun* должно меняться на ИСТИНА перед выполнением команды.
Рассмотрим пример программы, в которой выполняется специальная обработка, если значение флага *P_First_Run* равно ИСТИНА.

В таблице ниже показана взаимосвязь между значениями параметра *isFirstRun* и флага *P_First_Run*.
Работа флага *P_First_Run* зависит от того, активирована ли уже указанная программа или деактивирована.

Значение *isFirstRun*	Статус программы	Значение *P_First_Run*
ИСТИНА	Деактивирована.	Переходит в состояние ИСТИНА на один цикл выполнения задачи при выполнении программы. Возвращается в состояние ЛОЖЬ в следующем цикле выполнения задачи.
	Уже активирована.	Остается в состоянии ЛОЖЬ.
ЛОЖЬ	---	Остается в состоянии ЛОЖЬ.

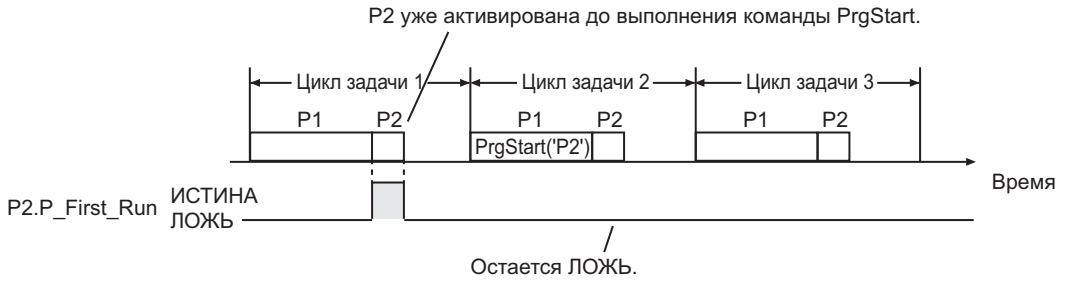
● Когда *isFirstRun* = ИСТИНА и программа деактивирована

Флаг *P_First_Run* переходит в состояние ИСТИНА, когда начинается выполнение программы, и остается в этом состоянии в течение одного цикла выполнения задачи. После этого значение флага *P_First_Run* меняется на ЛОЖЬ.



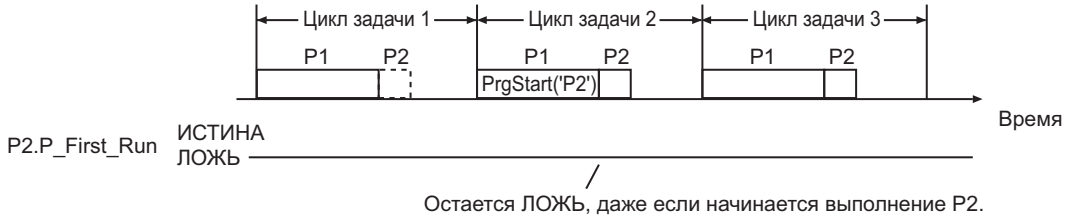
● Когда *isFirstRun* = ИСТИНА и программа уже активирована

Флаг *P_First_Run* остается в состоянии ЛОЖЬ, даже если выполняется команда PrgStart.



● Когда *isFirstRun* = ЛОЖЬ

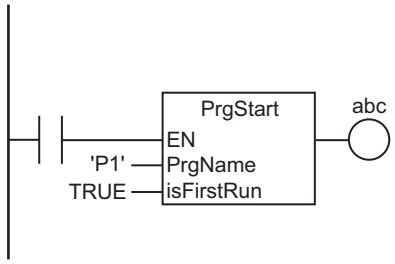
Флаг *P_First_Run* остается в состоянии ЛОЖЬ, даже когда начинается выполнение программы.



Пример записи

Ниже показан пример записи команды для активации программы с именем «P1».

LD



ST

```
abc:=PrgStart('P1', TRUE);
```

Связанные системные переменные

Имя	Значение	Тип данных	Описание
P_First_Run	Флаг первого цикла программы	BOOL	Этот флаг находится в состоянии «ИСТИНА» в течение одного цикла выполнения задачи после начала выполнения программы. Все остальное время он находится в состоянии ЛОЖЬ. В то же время, если значение *isFirstRun* поменяется на ЛОЖЬ и будет выполнена команда PrgStart, флаг P_First_Run останется в состоянии ЛОЖЬ даже после начала выполнения программы. Этот флаг можно использовать для выполнения специальной обработки в начале выполнения программы.
P_First_RunMode	Флаг первого цикла в режиме «Выполнение»	BOOL	Этот флаг находится в состоянии ИСТИНА только в течение одного цикла выполнения задачи после перехода модуля ЦПУ из режима «Программирование» в режим «Выполнение», если программа выполняется. Если программа не выполняется, этот флаг остается в состоянии ЛОЖЬ. Данный флаг можно использовать для выполнения инициализации в начале работы модуля ЦПУ.

Дополнительная информация

- Для деактивации указанной программы из программы пользователя используйте команду *PrgStop* на стр. 2-1032.
- Для чтения статуса указанной программы из программы пользователя используйте команду *PrgStatus* на стр. 2-1052.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды для уже активированной программы не приводит к возникновению ошибки.
- Если эта команда выполняется несколько раз для одной и той же программы, используется значение *isFirstRun*, заданное для экземпляра команды, который был выполнен первым.
- Если после выполнения команды PrgStart для той же программы будет выполнена команда PrgStop, причем это будет сделано до начала выполнения программы, эта программа выполнена не будет.
- Если после выполнения команды PrgStop для той же программы будет выполнена команда PrgStart, причем это будет сделано до наступления момента выполнения программы, эта программа не будет деактивирована.
- Работа программ (т. е. выполняется программа или не выполняется) непосредственно после перехода модуля ЦПУ в режим работы «Выполнение» определяется параметром *Initial Status (Начальное состояние)*, который задается для каждой программы в Sysmac Studio. Это означает, что команда PrgStart или PrgStop будет деактивирована после смены режима работы, если она была выполнена до смены режима работы.
- Если эта команда выполняется для программы, которая относится к другой задаче, то время выполнения указанной программы зависит от приоритета выполнения обеих задач. В некоторых случаях контроллер может работать непредсказуемым образом. Чтобы гарантировать выполнение указанной программы в том же цикле выполнения задачи, в котором выполняется

команда, команду можно выполнить в первой программе задачи, которой назначена и указанная программа.

- Внутренние переменные, входные переменные, выходные переменные и входные-выходные переменные указанной программы сохраняют значения, которые они содержали при предыдущем выполнении программы. Если значения этих переменных нужно инициализировать перед выполнением программы, поменяйте значение параметра *isFirstRun* на ИСТИНА и выполните команду, при этом предусмотрите выполнение процедуры инициализации в указанной программе, если значение флага *P_First_Run* = ИСТИНА.
- В указанном ниже случае произойдет ошибка. Выход *Out* перейдет в состояние ЛОЖЬ.
  а) Программы с указанным именем *PrgName* не существует.

Пример программы

Пример программы для команды PrgStop см. в разделе *Пример программы* на стр. 2-1040.

PrgStop

Команда PrgStop запрещает выполнение указанной программы.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
PrgStop	Деактивация программы	FUN	(@)PrgStop EN PrgName Out	Out:=PrgStop(PrgName);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
PrgName	Имя программы	Вход	Имя указанной программы	Макс. 128 байт (127 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	*1
Out	Флаг нормального завершения	Выход	Флаг нормального завершения ИСТИНА: нормальное завершение ЛОЖЬ: завершение с ошибкой	Зависит от типа данных.	---	---

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
PrgName																				OK
Out	OK																			

Функция

Команда PrgStop запрещает выполнение программы, указанной параметром *PrgName*.

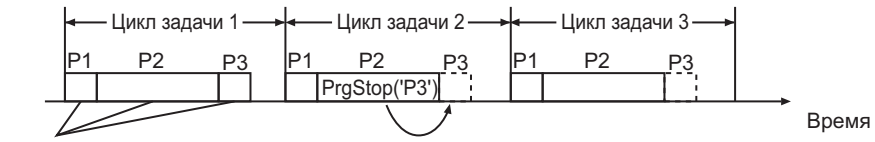
Указанная программа деактивируется, начиная со следующего раза, когда наступает время ее выполнения.
Даже если выполнение указанной программы уже запрещено, ошибки не возникает.
Указанная программа может находиться в той же задаче, что и эта команда, или в другой задаче.
Можно также указать программу, в которой находится сама команда. Если в команде указана программа, в которой находится сама команда, программа выполняется до конца в цикле задачи, в котором выполнена команда, и деактивируется, начиная со следующего цикла выполнения задачи.
Значение переменной *Out* равно ИСТИНА при нормальном завершении команды и ЛОЖЬ — при завершении команды с ошибкой.

Пример работы команды для случая, когда указана программа в текущей задаче

Ниже рассматривается пример работы команды для случая, когда в команде указана программа, которая находится в той же задаче, что и сама команда.

● Деактивация программы, выполняемой после команды PrgStop

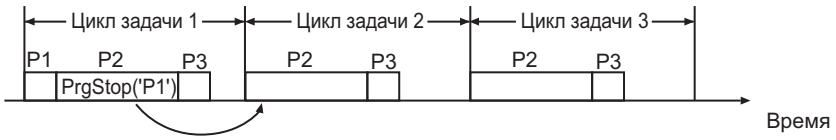
- В данном примере в одной задаче имеются три программы: P1, P2 и P3.
- Программа P3 выполняется в цикле выполнения задачи 1.
- В программе P2 цикла выполнения задачи 2 выполняется команда PrgStop, в которой указана программа P3.
- Программа P3 выполняется после программы P2, поэтому программа P3 деактивируется, начиная с цикла выполнения задачи 2.
- После этого программа P3 остается деактивированной, даже если команда PrgStop с указанной программой P3 больше не выполняется.



P3 выполняется после P2, поэтому P3 деактивируется в том же цикле задачи, в котором выполняется команда PrgStop.

● Деактивация программы, выполняемой до команды PrgStop

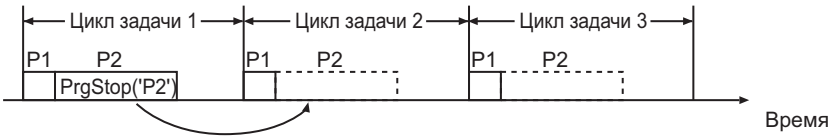
- В данном примере в одной задаче имеются три программы: P1, P2 и P3.
- Программа P1 выполняется в цикле выполнения задачи 1.
- В программе P2 цикла выполнения задачи 1 выполняется команда PrgStop, в которой указана программа P1.
- Программа P1 выполняется до программы P2, поэтому программа P1 деактивируется, начиная с цикла выполнения задачи 2.
- После этого программа P1 остается деактивированной, даже если команда PrgStop с указанной программой P1 больше не выполняется.



P1 выполняется перед P2, поэтому P1 деактивируется со следующего цикла, т. е. после цикла, в котором выполняется команда PrgStop.

● Деактивация программы, содержащей команду PrgStop

- В данном примере в одной задаче имеются две программы: P1 и P2.
- Программа P2 выполняется в цикле выполнения задачи 1.
- В программе P2 цикла выполнения задачи 1 выполняется команда PrgStop, в которой указана программа P2.
- В цикле задачи 1 программа P2 выполняется до конца.
- Программа P2 деактивируется, начиная с цикла выполнения задачи 2.
- После этого программа P2 остается деактивированной, даже если команда PrgStop с указанной программой P2 больше не выполняется.



Программа выполняется до конца в цикле задачи, в котором была выполнена команда PrgStop.

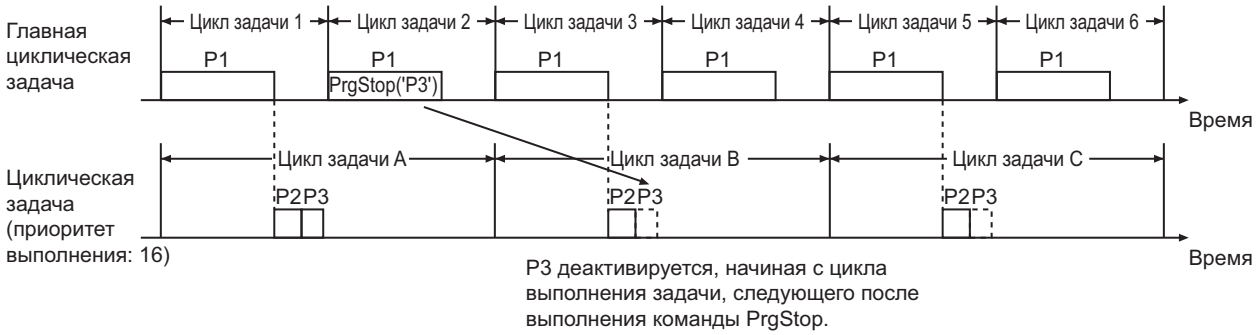
Программа деактивируется со следующего цикла, т. е. после цикла, в котором выполняется команда PrgStop.

Пример работы команды для случая, когда указана программа в другой задаче

Ниже рассматривается пример работы команды для случая, когда в команде указана программа, которая находится не в той же задаче, что и команда, а в другой задаче.

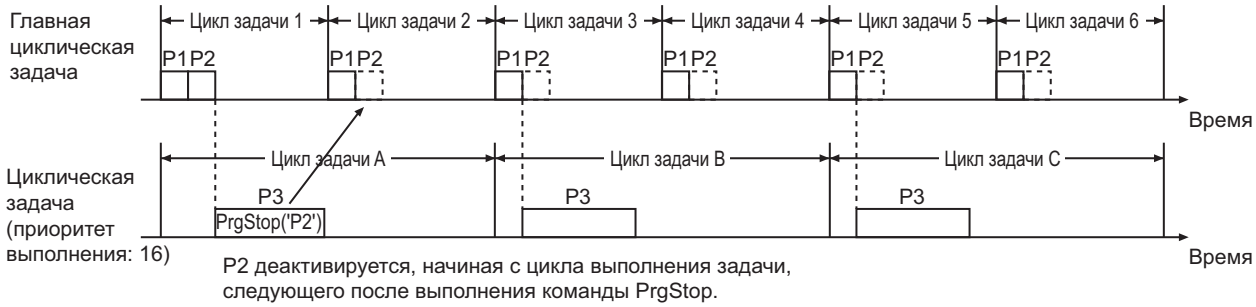
● Деактивация программы в задаче с более низким приоритетом, чем у текущей задачи

- В данном примере имеются три программы. Программа P1 находится в главной циклической задаче, а программы P2 и P3 находятся в циклической задаче.
- Программа P3 выполняется в цикле выполнения задачи A циклической задачи.
- В программе P1 цикла выполнения задачи 2 главной циклической задачи выполняется команда PrgStop, в которой указана программа P3.
- Программа P3 деактивируется, начиная с цикла выполнения задачи B циклической задачи, которая выполняется после выполнения команды PrgStop.
- После этого программа P3 остается деактивированной, даже если команда PrgStop с указанной программой P3 больше не выполняется.



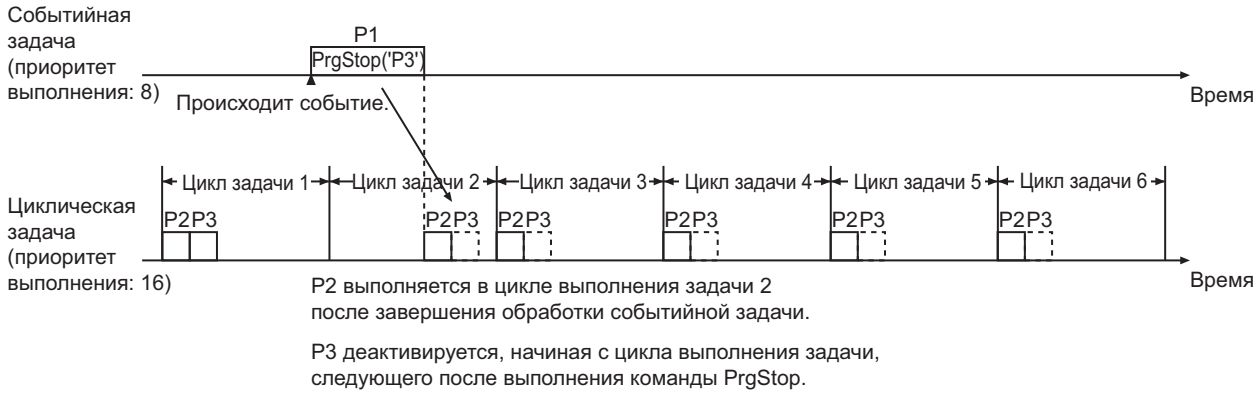
● Деактивация программы в задаче с более высоким приоритетом, чем у текущей задачи

- В данном примере имеются три программы. Программы P1 и P2 находятся в главной циклической задаче, а программа P3 находится в циклической задаче.
- Программа P2 выполняется в цикле выполнения задачи 1 главной циклической задачи.
- В программе P3 цикла выполнения задачи A циклической задачи выполняется команда PrgStop, в которой указана программа P2.
- Программа P2 деактивируется, начиная с цикла выполнения задачи 2 главной циклической задачи, которая выполняется после выполнения команды PrgStop.
- После этого программа P2 остается деактивированной, даже если команда PrgStop с указанной программой P2 больше не выполняется.



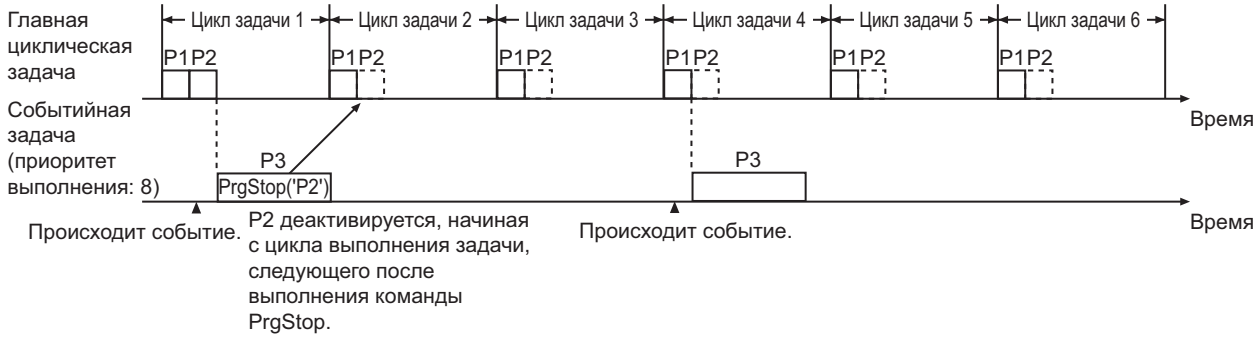
● Деактивация программы в задаче с более низким приоритетом из событийной задачи

- В данном примере имеются три программы. Программа P1 находится в событийной задаче (приоритет выполнения: 8), а программы P2 и P3 находятся в циклической задаче (приоритет выполнения: 16).
- Программа P3 выполняется в цикле выполнения задачи 1 циклической задачи.
- В событийной задаче выполняется команда PrgStop, в которой указана программа P3.
- Когда выполняется событийная задача, в цикле выполнения задачи 2 циклической задачи после завершения обработки событийной задачи выполняются программы P2 и P3.
- В результате программа P3 деактивируется уже в цикле выполнения задачи 2 циклической задачи, так как она следует после выполнения команды PrgStop.
- После этого программа P3 остается деактивированной, даже если команда PrgStop с указанной программой P3 больше не выполняется.



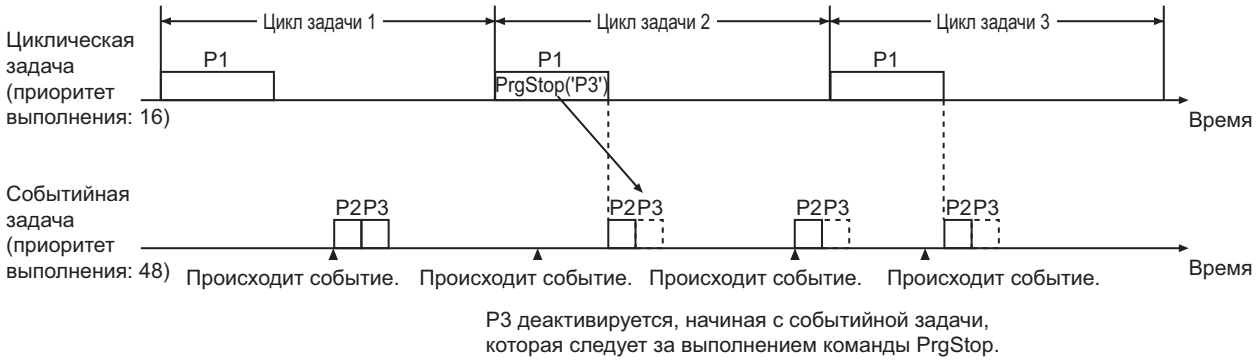
● Деактивация программы в задаче с более высоким приоритетом из событийной задачи

- В данном примере имеются три программы. Программы P1 и P2 находятся в главной циклической задаче, а программа P3 находится в событийной задаче.
- Программа P2 выполняется в цикле выполнения задачи 1 главной циклической задачи.
- В событийной задаче выполняется команда PrgStop, в которой указана программа P2.
- Программа P2 деактивируется, начиная с цикла выполнения задачи 2 главной циклической задачи, которая выполняется после выполнения команды PrgStop.
- После этого программа P2 остается деактивированной, даже если команда PrgStop с указанной программой P2 больше не выполняется.



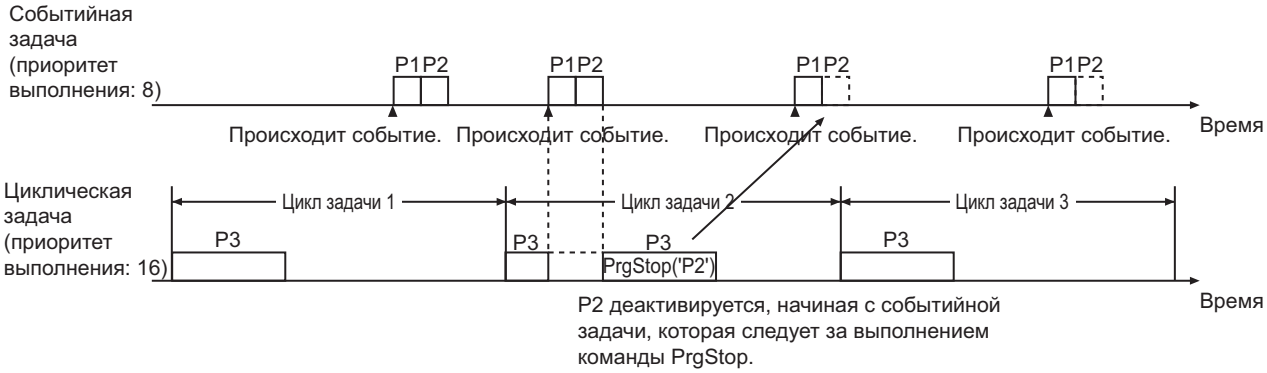
● Деактивация программы в событийной задаче с более низким приоритетом из циклической задачи

- В данном примере имеются три программы. Программа P1 находится в циклической задаче (приоритет выполнения: 16), а программы P2 и P3 находятся в событийной задаче (приоритет выполнения: 48).
- Программа P3 выполняется в событийной задаче.
- В циклической задаче выполняется команда PrgStop, в которой указана программа P3.
- Программа P3 в событийной задаче деактивируется, начиная с событийной задачи, которая выполняется после выполнения команды PrgStop.
- После этого программа P3 остается деактивированной, даже если команда PrgStop с указанной программой P3 больше не выполняется.



● Деактивация программы в событийной задаче с более высоким приоритетом из циклической задачи

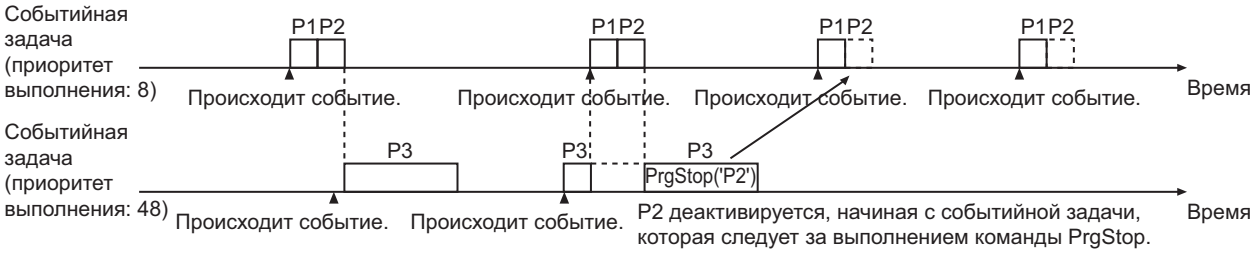
- В данном примере имеются три программы. Программы P1 и P2 находятся в событийной задаче (приоритет выполнения: 8), а программа P3 находится в циклической задаче (приоритет выполнения: 16).
- Программа P2 выполняется в событийной задаче.
- В циклической задаче выполняется команда PrgStop, в которой указана программа P2.
- Программа P2 в событийной задаче деактивируется, начиная с событийной задачи, которая выполняется после выполнения команды PrgStop.
- После этого программа P2 остается деактивированной, даже если команда PrgStop с указанной программой P2 больше не выполняется.



● Деактивация программы в событийной задаче с более низким приоритетом из другой событийной задачи

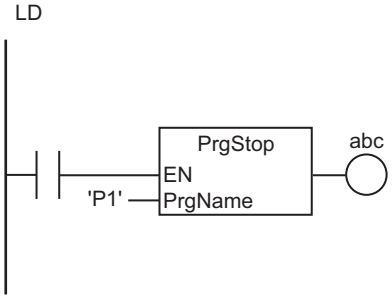
- В данном примере имеются три программы. Программа P1 находится в событийной задаче (приоритет выполнения: 8), а программы P2 и P3 находятся в событийной задаче (приоритет выполнения: 48).
- Программа P3 в событийной задаче (приоритет выполнения: 48) выполняется.
- В событийной задаче (приоритет выполнения: 8) выполняется команда PrgStop, в которой указана программа P3.
- Программа P3 в событийной задаче (приоритет выполнения: 48) деактивируется, начиная с событийной задачи (приоритет выполнения: 48), которая выполняется после выполнения команды PrgStop.
- После этого программа P3 остается деактивированной, даже если команда PrgStop с указанной программой P3 больше не выполняется.

- **Деактивация программы в событийной задаче с более высоким приоритетом из другой событийной задачи**
  - В данном примере имеются три программы. Программы P1 и P2 находятся в событийной задаче (приоритет выполнения: 8), а программа P3 находится в событийной задаче (приоритет выполнения: 48).
  - Программа P2 в событийной задаче (приоритет выполнения: 8) выполняется.
  - В событийной задаче (приоритет выполнения: 48) выполняется команда PrgStop, в которой указана программа P2.
  - Программа P2 в событийной задаче (приоритет выполнения: 8) деактивируется, начиная с событийной задачи (приоритет выполнения: 8), которая выполняется после выполнения команды PrgStop.
  - После этого программа P2 остается деактивированной, даже если команда PrgStop с указанной программой P2 больше не выполняется.



Пример записи

Ниже показан пример записи команды для деактивации программы с именем «P1».



Дополнительная информация

- Для активации указанной программы из программы пользователя используйте команду *PrgStart* на стр. 2-1022.
- Для чтения статуса указанной программы из программы пользователя используйте команду *PrgStatus* на стр. 2-1052.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды для уже деактивированной программы не приводит к возникновению ошибки.
- Если команда PrgStop будет выполнена после выполнения команды PrgStart для той же программы, причем это будет сделано до начала выполнения программы, эта программа выполнена не будет.
- Если после выполнения команды PrgStop для той же программы будет выполнена команда PrgStart, причем это будет сделано до наступления момента выполнения программы, эта программа не будет деактивирована.
- Команды, имеющие входную переменную *Execute*, продолжают выполняться до полного завершения, даже если время их выполнения превышает длительность одного цикла задачи. Прежде чем деактивировать программу, в которой имеется команда такого типа, необходимо убедиться, что переменная *Busy* данной команды содержит значение ЛОЖЬ, то есть команда не выполняется в данный момент.
- Для выполнения команды NX_DOutTimeStamp или NX_AryDOutTimeStamp может потребоваться несколько циклов выполнения задачи. Прежде чем деактивировать программу, содержащую такую команду, необходимо убедиться, что переменная *Enable* данной команды содержит значение ЛОЖЬ.
- Работа программ (т. е. выполняется программа или не выполняется) непосредственно после перехода модуля ЦПУ в режим работы «Выполнение» определяется параметром *Initial Status (Начальное состояние)*, который задается для каждой программы в Sysmac Studio. Это означает, что команда PrgStart или PrgStop будет деактивирована после смены режима работы, если она была выполнена до смены режима работы.
- Если эта команда выполняется для программы, которая относится к другой задаче, то время деактивации указанной программы зависит от приоритета выполнения обеих задач. В некоторых случаях контроллер может работать непредсказуемым образом. Чтобы гарантировать деактивацию указанной программы в том же цикле выполнения задачи, в котором выполняется команда, команду можно выполнить в первой программе задачи, которой назначена и указанная программа.
- Перед выполнением данной команды необходимо проверять, что для указанной программы соблюдаются следующие условия:
  a) В данный момент не выполняется какая-либо команда управления движением.
  b) В данный момент не выполняется какая-либо команда, имеющая входную переменную *Execute*, т. е. команда, которая продолжает выполняться до полного завершения, даже если время ее выполнения превышает длительность одного цикла задачи.
  c) Отсутствуют команды с меткой времени, которые ожидают наступления указанного времени.
- При деактивации указанной программы ее выходные переменные не сбрасываются. В переменных сохраняются значения, которые они содержали перед выполнением деактивации. Если необходимо, чтобы выходы деактивируемой программы сбрасывались, используйте в указанной программе функцию главного управления, чтобы заранее сбрасывать выходные переменные.

- Даже если программа будет деактивирована с помощью этой команды, любые содержащиеся в ней команды функционального блока, имеющие входную переменную *Execute*, продолжат выполняться и будут выполнены до конца.
- Даже если программа будет деактивирована с помощью этой команды, любые содержащиеся в ней команды управления движением продолжат выполняться и будут выполнены до конца.
- В указанном ниже случае произойдет ошибка. Выход *Out* перейдет в состояние ЛОЖЬ.
  a) Программы с указанным именем *PrgName* не существует.

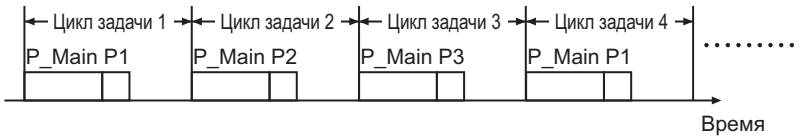
Пример программы

В данном разделе будут рассмотрены два примера программ для объяснения работы команды.

Пример последовательного выполнения программ

Ниже представлен пример, в котором три программы выполняются одна за другой, по одной программе в каждом цикле выполнения задачи.

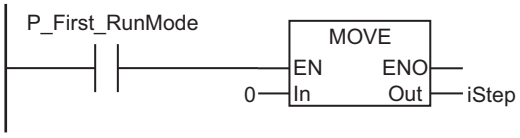
В примере используются программы с именами P1, P2 и P3.
Эти программы выполняются по кругу, последовательно, одна за другой, по одной программе в каждом цикле выполнения задачи.
Кроме того, имеется программа P_Main, которая включает команды для активации или деактивации этих трех программ.



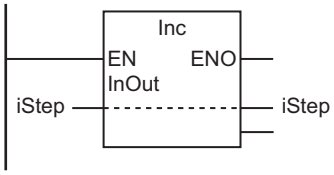
● Программа на языке LD

Переменная	Тип данных	По умолчанию	Комментарий
iStep	DINT	0	Номер программы для выполнения

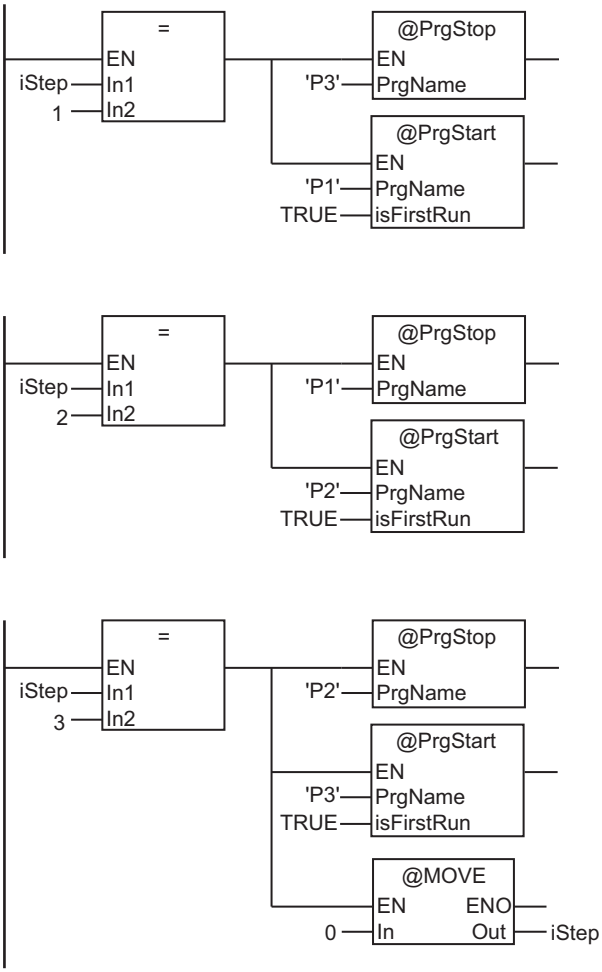
Присвойте 0 переменной ***iStep*** в начале работы.



Увеличение переменной ***iStep*** на 1.



Выполнение команд PrgStop и PrgStart.



● Программа на языке ST

Переменная	Тип данных	По умолчанию	Комментарий
iStep	DINT	0	Номер программы для выполнения

```
// Обнуление переменной iStep в начале работы.
IF P_First_RunMode THEN
  iStep:=0;
END_IF;

// Увеличение переменной iStep на 1.
iStep:=iStep+1;

// Выполнение команд PrgStop и PrgStart.
IF iStep = 1 THEN
  PrgStop('P3');
  PrgStart('P1',TRUE);
ELSIF iStep = 2 THEN
  PrgStop('P1');
  PrgStart('P2',FALSE);
ELSIF iStep = 3 THEN
```

```
  PrgStop('P2');
  PrgStart('P3',TRUE);
  iStep:=0;
END_IF;
```

Выполнение указанных программ при следующем запуске

В данном примере рассматривается случай, когда требуется указать несколько программ для выполнения при следующем запуске контроллера.

Программы, которые должны быть выполнены при следующем запуске контроллера, необходимо указать до отключения питания контроллера.
При следующем включении источника питания будут выполняться только указанные программы.

● Программы, модули и конфигурация модулей

Имеется восемь программ: программа 1, программа 2,... программа 8.
Также имеется пять модулей: модуль A, модуль B,... модуль E. Каждая программа относится к одному из этих пяти модулей.

Модуль	Программы в модуле
Модуль A	Программа 1
Модуль B	Программа 2
Модуль C	Программа 3 и программа 4
Модуль D	Программа 5, программа 6 и программа 7
Модуль E	Программа 8

Программы, которые должны выполняться, указываются путем указания соответствующего модуля. Комбинация выполняемых модулей называется конфигурацией модулей.
Например, если будет указана конфигурация модулей для выполнения модулей A и C, то будут выполняться программы 1, 3 и 4.

● Указание конфигураций модулей для выполнения

Конфигурации модулей указываются в виде текстовых данных в конфигурационном файле. Конфигурационный файл носит имя Config.txt и хранится в корневом каталоге на карте памяти SD.
Конфигурационный файл может содержать несколько конфигураций модулей.
Перед выключением питания контроллера оператор с помощью сенсорной панели указывает конфигурацию модулей (одну из тех, что содержатся в конфигурационном файле) для выполнения при следующем запуске контроллера.

● Формат конфигурационного файла

Формат конфигурационного файла приведен в следующей таблице.

Строка	Содержание
Строка 1	Количество конфигураций модулей
Строка 2 и последующие строки	Номер конфигурации модулей, флаг выполнения модуля A*1, флаг выполнения модуля B, флаг выполнения модуля C, флаг выполнения модуля D, флаг выполнения модуля E

*1. Модуль выполняется, если флаг равен TRUE (ИСТИНА), и не выполняется, если флаг равен FALSE (ЛОЖЬ).

Ниже приведен пример содержимого конфигурационного файла.

```
3
Config1, TRUE, TRUE, TRUE, FALSE, FALSE
Config2, TRUE, TRUE, FALSE, TRUE, FALSE
Config3, TRUE, TRUE, TRUE, FALSE, TRUE
```

Этот конфигурационный файл содержит три конфигурации: Config1, Config2 и Config3. Например, в конфигурации модулей Config1 указано, что выполняются модули A, B и C, а модули D и E не выполняются.

● Определения типов данных

Определение структуры с именем *myConfig* показано в таблице ниже.

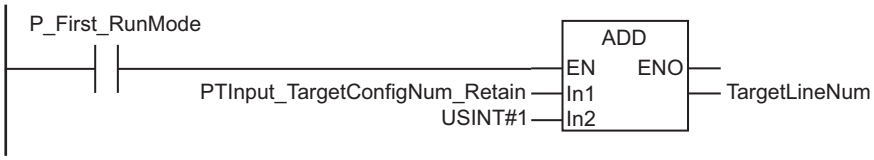
Структура		Переменная	Тип данных	Тип смещения	Комментарий
	▼	myConfig	STRUCT	NJ	Конфигурация модулей
		configName	STRING[32]		Имя конфигурации модулей
		moduleA	BOOL		Флаг выполнения модуля A
		moduleB	BOOL		Флаг выполнения модуля B
		moduleC	BOOL		Флаг выполнения модуля C
		moduleD	BOOL		Флаг выполнения модуля D
		moduleE	BOOL		Флаг выполнения модуля E

● Программа на языке LD

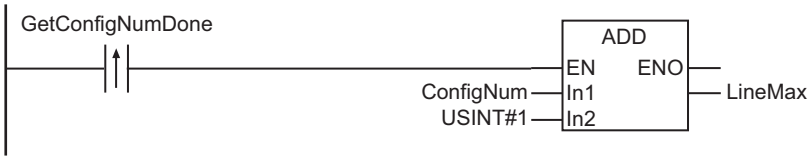
Переменная	Тип данных	По умолчанию	Сохранение	Комментарий
Open	FileOpen		☐	Экземпляр команды FileOpen
TopLineGetter	FileGets		☐	Экземпляр команды FileGets
LineGetter	FileGets		☐	Экземпляр команды FileGets
Close	FileClose		☐	Экземпляр команды FileClose
PTInput_TargetConfigNum_Retain	USINT	0	☑	Номер конфигурации модулей для выполнения при следующем запуске работы
CurrentLineNum	USINT	1	☐	Текущая строка конфигурационного файла
TargetLineNum	USINT	0	☐	Строка для *CurrentConfig* в конфигурационном файле
ConfigNum	USINT	1	☐	Число, указанное в строке 1 конфигурационного файла
LineMax	USINT	3	☐	Количество строк в конфигурационном файле, полученное по значению *ConfigNum*
isOverLine	BOOL	ЛОЖЬ	☐	Флаг ошибки в случае, когда значение *PTInput_TargetConfigNum_Retain* больше значения *LineMax*
Busy	BOOL	ЛОЖЬ	☐	Флаг обработки
SubDeliNG	BOOL	ЛОЖЬ	☐	Флаг завершения чтения с ошибкой для *CurrentConfig*
Error	BOOL	ЛОЖЬ	☐	Флаг ошибки

Переменная	Тип данных	По умолчанию	Сохранение	Комментарий
opening	BOOL	ЛОЖЬ	☐	Флаг выполнения открытия конфигурационного файла
myFileID	DWORD	0	☐	Идентификатор конфигурационного файла
TopLineGetting	BOOL	ЛОЖЬ	☐	Флаг выполнения чтения *ConfigNum*
GetConfigNumDone	BOOL	ЛОЖЬ	☐	Флаг завершения чтения *ConfigNum*
SelectDone	BOOL	ЛОЖЬ	☐	Флаг завершения чтения *CurrentConfig*
reading	BOOL	ЛОЖЬ	☐	Флаг выполнения чтения строки 2 или строки с более высоким номером из конфигурационного файла
CurrentConfig	myConfig	(configName:='', moduleA:=FALSE, moduleB:=FALSE, moduleC:=FALSE, moduleD:=FALSE)	☐	Конфигурация модулей для выполнения при следующем запуске работы
Error_exceptOpen	BOOL	ЛОЖЬ	☐	Флаг выполнения закрытия конфигурационного файла при возникновении ошибки

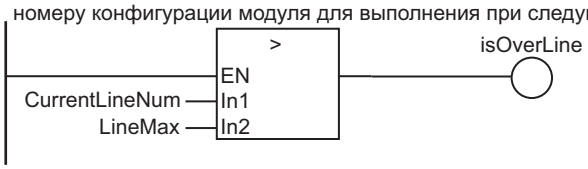
Получение номера конфигурации модуля для выполнения при следующем запуске работы.



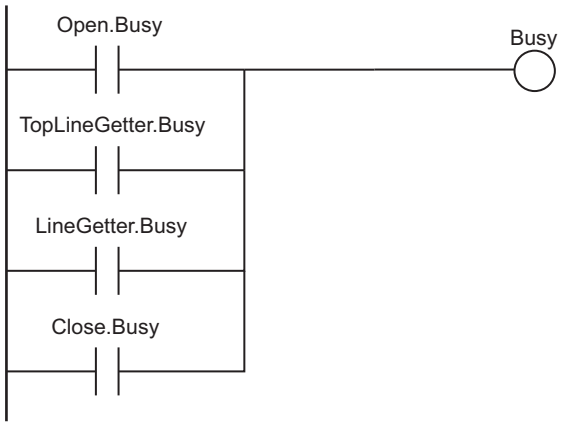
Расчет количества строк по содержимому строки 1 файла конфигурации.

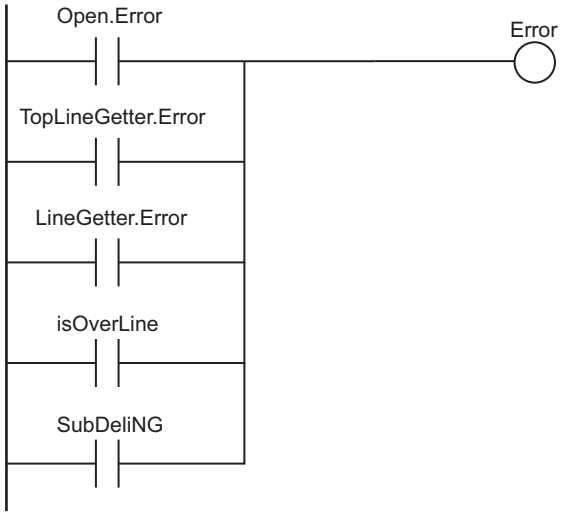


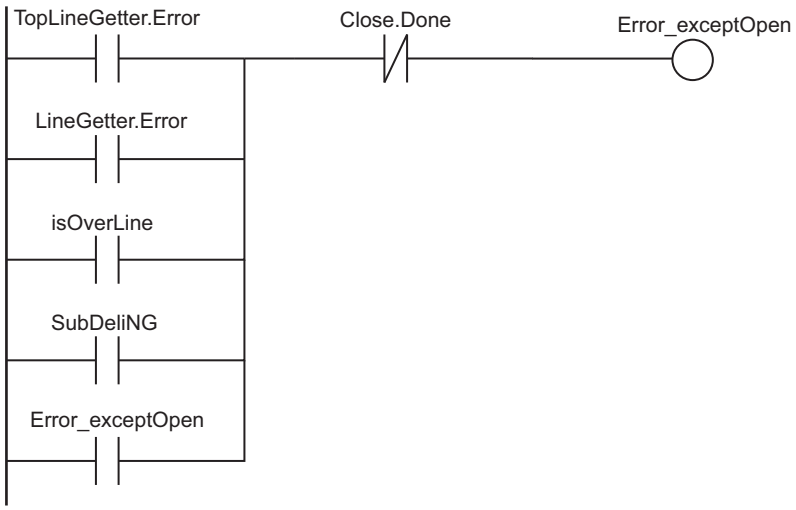
Обнаружение ошибки, когда количество строк в файле конфигурации не соответствует номеру конфигурации модуля для выполнения при следующем запуске работы.



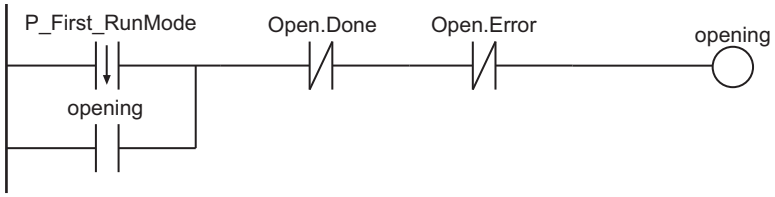
Обработка флага обработки и флагов ошибок.

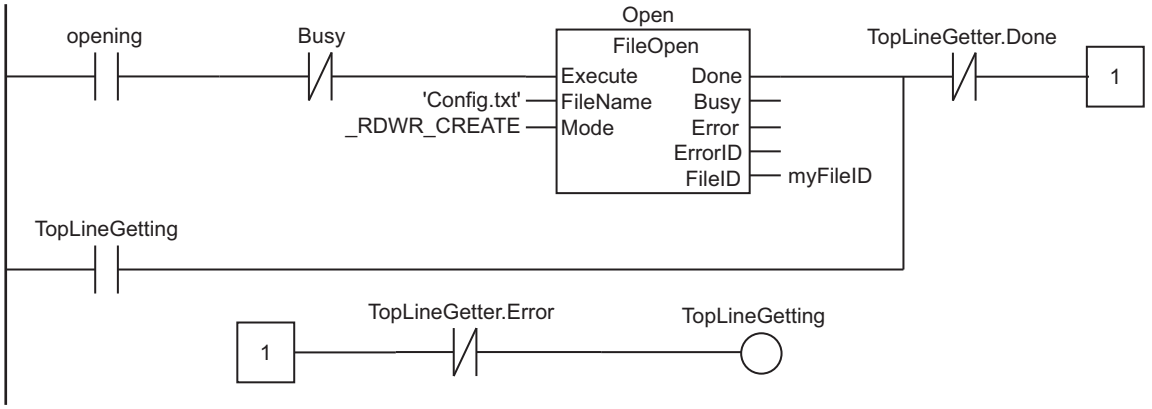




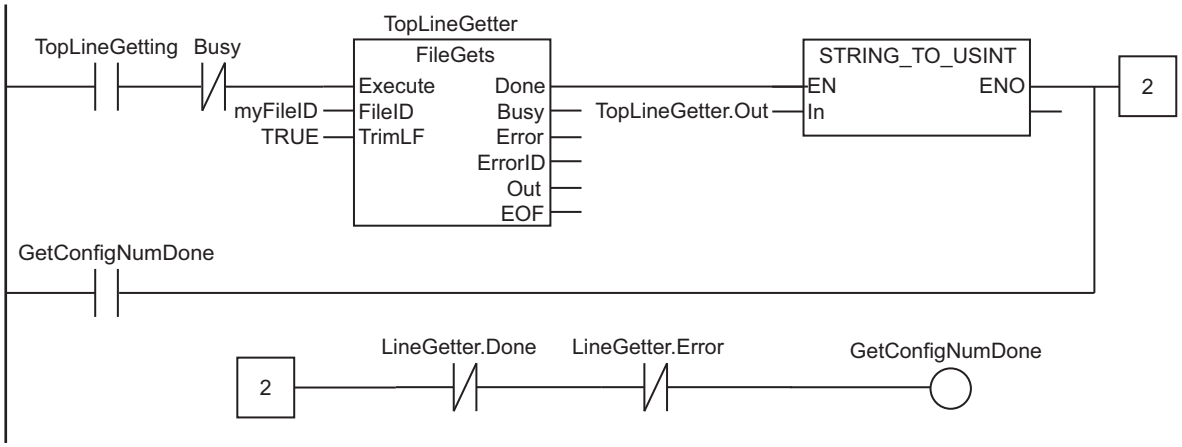


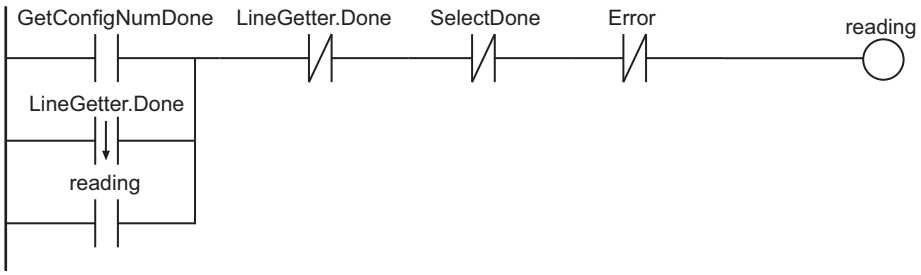
Открытие файла конфигурации.



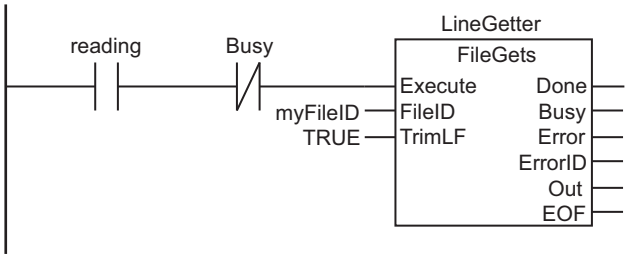


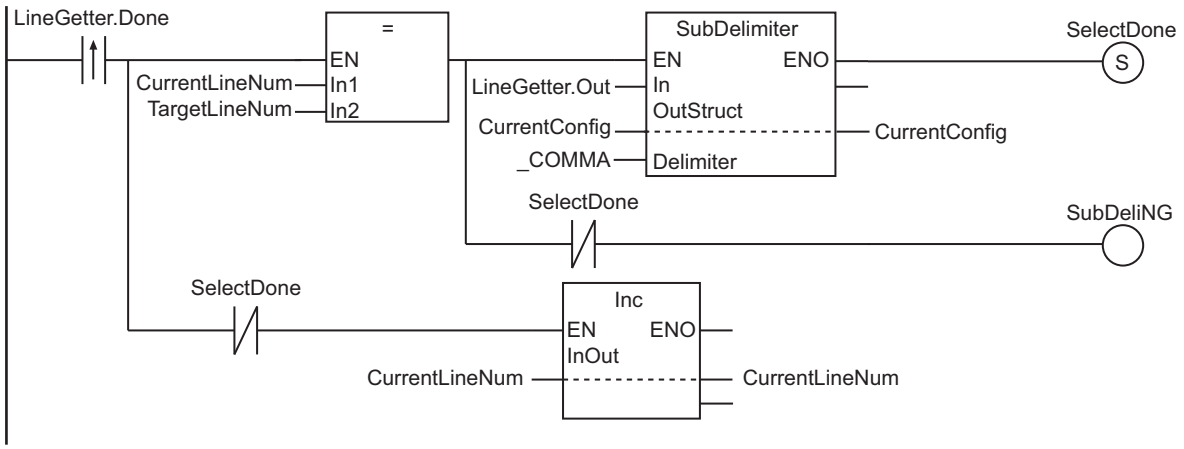
Чтение строки 1 файла конфигурации.



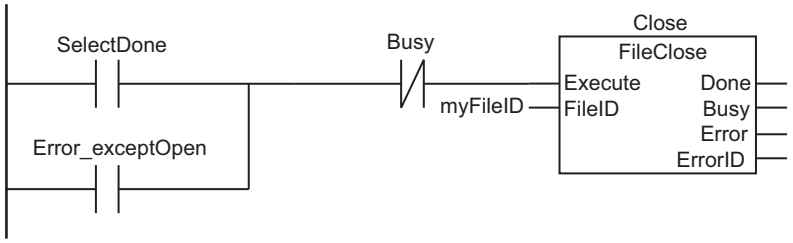


Чтение строки 2 или больше из файла конфигурации.

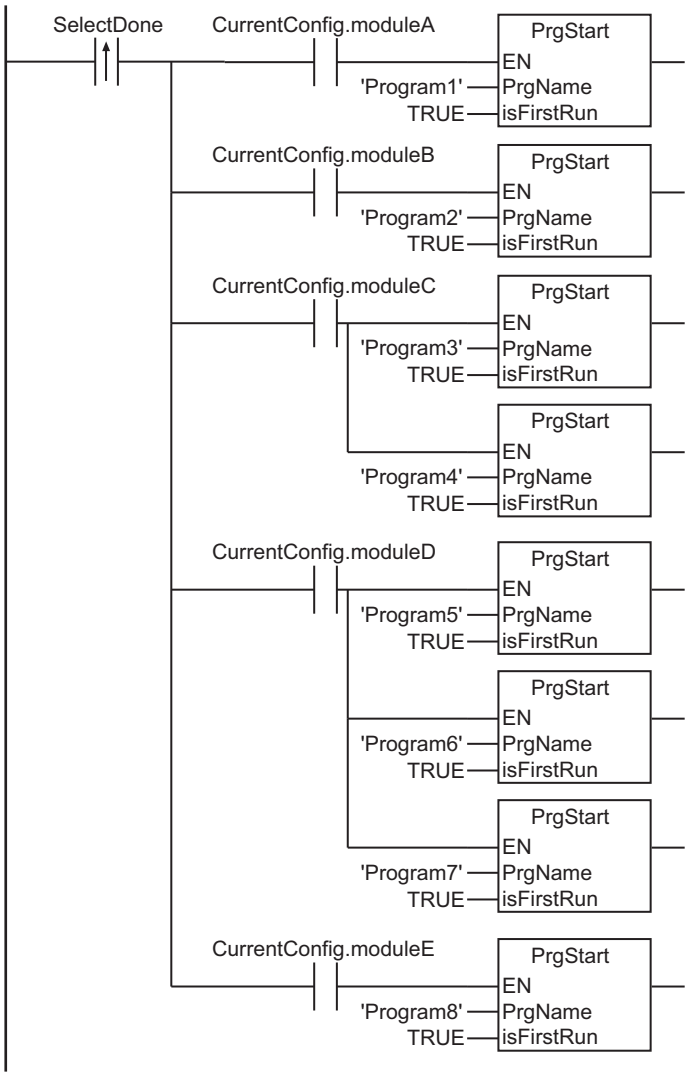




Закрытие файла конфигурации.



Выполнение команды PrgStart.



● Программа на языке ST

Переменная	Тип данных	По умолчанию	Сохранение	Комментарий
Open	FileOpen		☐	Экземпляр команды FileOpen
TopLineGetter	FileGets		☐	Экземпляр команды FileGets
LineGetter	FileGets		☐	Экземпляр команды FileGets
Close	FileClose		☐	Экземпляр команды FileClose
PTInput_TargetConfigNum_Retain	USINT	0	☑	Номер конфигурации модулей для выполнения при следующем запуске работы
CurrentLineNum	USINT	1	☐	Текущая строка конфигурационного файла
TargetLineNum	USINT	0	☐	Строка для *CurrentConfig* в конфигурационном файле
ConfigNum	USINT	1	☐	Число, указанное в строке 1 конфигурационного файла

Переменная	Тип данных	По умолчанию	Сохранение	Комментарий
LineMax	USINT	3	☐	Количество строк в конфигурационном файле, полученное по значению *ConfigNum*
isOverLine	BOOL	ЛОЖЬ	☐	Флаг ошибки в случае, когда значение *PTInput_TargetConfigNum_Retain* больше значения *LineMax*
Busy	BOOL	ЛОЖЬ	☐	Флаг обработки
SubDeliNG	BOOL	ЛОЖЬ	☐	Флаг завершения чтения с ошибкой для *CurrentConfig*
Error	BOOL	ЛОЖЬ	☐	Флаг ошибки
opening	BOOL	ЛОЖЬ	☐	Флаг выполнения открытия конфигурационного файла
myFileID	DWORD	0	☐	Идентификатор конфигурационного файла
TopLineGetting	BOOL	ЛОЖЬ	☐	Флаг выполнения чтения *ConfigNum*
GetConfigNumDone	BOOL	ЛОЖЬ	☐	Флаг завершения чтения *ConfigNum*
SelectDone	BOOL	ЛОЖЬ	☐	Флаг завершения чтения *CurrentConfig*
reading	BOOL	ЛОЖЬ	☐	Флаг выполнения чтения строки 2 или строки с более высоким номером из конфигурационного файла
CurrentConfig	myConfig	(configName:='', moduleA:=FALSE, moduleB:=FALSE, moduleC:=FALSE, moduleD:=FALSE)	☐	Конфигурация модулей для выполнения при следующем запуске работы
Error_exceptOpen	BOOL	ЛОЖЬ	☐	Флаг выполнения закрытия конфигурационного файла при возникновении ошибки
R_GetConfigNumDone	R_TRIG		☐	Экземпляр команды R_TRIG
RS_1	RS		☐	Экземпляр команды RS
RS_2	RS		☐	Экземпляр команды RS
SecondCycle	F_TRIG		☐	Экземпляр команды F_TRIG
RS_3	RS		☐	Экземпляр команды RS
ConvertDone	BOOL	ЛОЖЬ	☐	Флаг «Преобразование выполнено» для преобразования символа в строке 1 конфигурационного файла в число.
RS_4	RS		☐	Экземпляр команды RS
F_LineGetterDone	F_TRIG		☐	Экземпляр команды F_TRIG
R_LineGetterDone	R_TRIG		☐	Экземпляр команды R_TRIG

Переменная	Тип данных	По умолчанию	Сохранение	Комментарий
isTargetLine	BOOL	ЛОЖЬ	☐	Флаг, указывающий, что текущая строка является строкой конфигурации модулей для выполнения при следующем запуске работы
SubDeliCondition	BOOL	ЛОЖЬ	☐	Флаг выполнения расширения конфигурации модулей до *CurrentConfig*
RS_5	RS		☐	Экземпляр команды RS
SubDeliDone	BOOL	ЛОЖЬ	☐	Флаг завершения расширения конфигурации модулей до *CurrentConfig*
R_SelectDone	R_TRIG		☐	Экземпляр команды R_TRIG

```
// Получение номера конфигурации модулей для выполнения при следующем запуске работ
ы.
IF P_First_RunMode THEN
  TargetLineNum := PTInput_TargetConfigNum_Retain + USINT#1;
END_IF;

// Расчет количества строк по содержимому строки 1 конфигурационного файла.
R_GetConfigNumDone(Clk:=GetConfigNumDone);
IF R_GetConfigNumDone.Q THEN
  LineMax := ConfigNum + USINT#1;
END_IF;

// Обнаружение ошибки, когда количество строк в конфигурационном файле не соответст
вует номеру конфигурации модулей для выполнения при следующем запуске работы.
isOverLine := (CurrentLineNum > LineMax);

// Обработка флага обработки и флагов ошибок.
Busy := Open.Busy OR TopLineGetter.Busy OR LineGetter.Busy OR Close.Busy;

Error := Open.Error OR TopLineGetter.Error OR LineGetter.Error OR isOverLine OR Sub
DeliNG;

RS_1(Set:= (TopLineGetter.Error OR LineGetter.Error OR isOverLine OR SubDeliNG), re
set1 := Close.Done, Q1 => Error_exceptOpen);

// Открытие конфигурационного файла.
SecondCycle(Clk:=P_First_RunMode);
RS_2(Set := SecondCycle.Q, reset1:=(Open.Done OR Open.Error), Q1 => opening);
Open(Execute:=(opening & NOT(Busy)), FileName :='Config.txt', FileID => myFileID);
RS_3(Set := Open.Done, Reset1:=(TopLineGetter.Done OR TopLineGetter.Error), Q1=>Top
LineGetting);

// Чтение строки 1 конфигурационного файла
TopLineGetter(Execute :=(TopLineGetting & NOT(Busy)), FileID := myFileID, TrimLF :=
```

```
 TRUE);
ConfigNum := STRING_TO_USINT(EN:= TopLineGetter.Done, IN:=TopLineGetter.Out, ENO=>C
onvertDone);
RS_4(Set := ConvertDone, Reset1:=(LineGetter.Done OR LineGetter.Error), Q1=>GetConf
igNumDone);
F_LineGetterDone(Clk:=LineGetter.Done);
RS_5(Set := (GetConfigNumDone OR F_LineGetterDone.Q), Reset1:=(LineGetter.Done OR S
electDone OR Error), Q1=>reading);

// Чтение строки 2 или строки с большим номером из конфигурационного файла.
LineGetter(Execute:=(reading & NOT(Busy)), FileID:=myFileID, TrimLF := TRUE);
R_LineGetterDone(Clk:=LineGetter.Done);
isTargetLine := (CurrentLineNum = TargetLineNum);
SubDeliCondition := (R_LineGetterDone.Q & isTargetLine);
SubDelimiter(EN := SubDeliCondition, In := LineGetter.Out, OutStruct := CurrentConf
ig, Delimiter := _COMMA, ENO => SubDeliDone);
IF SubDeliDone THEN
  SelectDone := TRUE;
END_IF;
SubDeliNG := (SubDeliCondition & NOT(SubDeliDone));
Inc(EN := (R_LineGetterDone.Q & NOT(SelectDone)), InOut:= CurrentLineNum);

// Закрытие конфигурационного файла.
Close(Execute := ((SelectDone OR Error_exceptOpen) & NOT(Busy)), FileID := myFileID
);

// Выполнение команды PrgStart.
R_SelectDone(Clk:=SelectDone);
//модуль A
PrgStart(EN := (R_SelectDone.Q & CurrentConfig.moduleA), PrgName :='Program1', isFi
rstRun:=TRUE);
//модуль B
PrgStart(EN := (R_SelectDone.Q & CurrentConfig.moduleB), PrgName :='Program2', isFi
rstRun:=TRUE);
//модуль C
PrgStart(EN := (R_SelectDone.Q & CurrentConfig.moduleC), PrgName :='Program3', isFi
rstRun:=TRUE);
PrgStart(EN := (R_SelectDone.Q & CurrentConfig.moduleC), PrgName :='Program4', isFi
rstRun:=TRUE);
//модуль D
PrgStart(EN := (R_SelectDone.Q & CurrentConfig.moduleD), PrgName :='Program5', isFi
rstRun:=TRUE);
PrgStart(EN := (R_SelectDone.Q & CurrentConfig.moduleD), PrgName :='Program6', isFi
rstRun:=TRUE);
PrgStart(EN := (R_SelectDone.Q & CurrentConfig.moduleD), PrgName :='Program7', isFi
rstRun:=TRUE);
//модуль E
PrgStart(EN := (R_SelectDone.Q & CurrentConfig.moduleE), PrgName :='Program8', isFi
rstRun:=TRUE);
```

PrgStatus

Команда PrgStatus производит чтение статуса указанной программы.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
PrgStatus	Чтение статуса программы	FUN	(@)PrgStatus EN ENO PrgName Out	Out:=PrgStatus(PrgName);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
PrgName	Имя программы	Вход	Имя указанной программы	Макс. 128 байт (127 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	*1
Out	Статус программы	Выход	Статус программы в следующий раз, когда наступит время ее выполнения ИСТИНА: активирована. ЛОЖЬ: деактивирована.	Зависит от типа данных.	---	---

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
PrgName																				OK
Out	OK																			

Функция

Команда PrgStatus производит чтение статуса, который программа, указанная параметром *PrgName*, будет иметь в следующий раз, когда наступит время для ее выполнения.
Если указанная программа будет активирована, когда в следующий раз наступит время ее выполнения, значение переменной *Out* будет равно ИСТИНА.
Если указанная программа будет деактивирована, когда в следующий раз наступит время ее выполнения, значение переменной *Out* будет равно ЛОЖЬ.

В приведенной ниже таблице поясняется, что означают статусы «активирована» и «деактивирована» в следующий раз, когда наступает время выполнения программы.

Статус программы	Описание
Будет активирована в следующий раз, когда наступит время ее выполнения	• Параметр *Initial Status (Начальное состояние)* для соответствующей программы установлен равным **Run (Выполнение)** в Sysmac Studio. • Для программы была выполнена команда PrgStart.
Будет деактивирована в следующий раз, когда наступит время ее выполнения	• Параметр *Initial Status (Начальное состояние)* для соответствующей программы установлен равным **Stop (Стоп)** в Sysmac Studio. • Для программы была выполнена команда PrgStop.

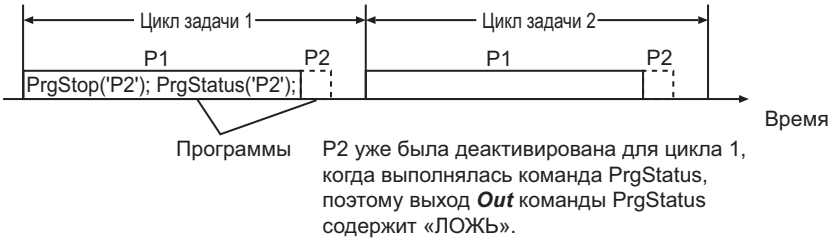
Указанная программа может находиться в той же задаче, что и эта команда, или в другой задаче.

Примеры работы команды

В данном разделе приведены некоторые примеры работы этой команды.

● Чтение статуса программы после команды PrgStatus в текущей задаче

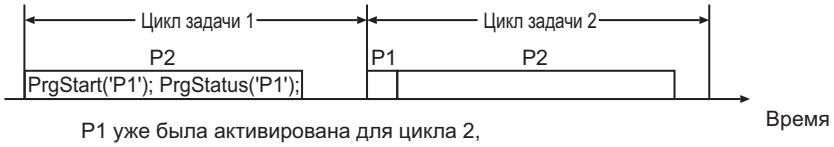
- В данном примере в одной задаче имеются две программы: P1 и P2.
- В программе P1 цикла выполнения задачи 1 выполняется команда PrgStop, в которой указана программа P2.
- Затем в программе P1 цикла выполнения задачи 1 выполняется команда PrgStatus, в которой указана программа P2.
- Программа P2 уже деактивирована в цикле 1, поэтому выход *Out* команды PrgStatus содержит ЛОЖЬ.



● Чтение статуса программы до команды PrgStatus в текущей задаче

- В данном примере в одной задаче имеются две программы: P1 и P2.
- В программе P2 цикла выполнения задачи 1 выполняется команда PrgStart, в которой указана программа P1.
- Затем в программе P2 цикла выполнения задачи 1 выполняется команда PrgStatus, в которой указана программа P1.

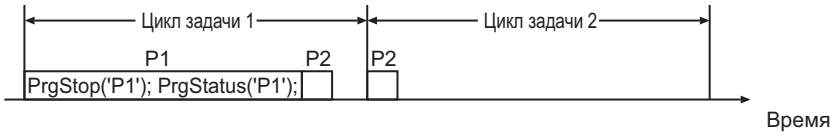
- Программа P1 была активирована для цикла выполнения задачи 2, поэтому выход *Out* команды PrgStatus содержит ИСТИНА.



P1 уже была активирована для цикла 2, когда выполнялась команда PrgStatus, поэтому выход ***Out*** команды PrgStatus содержит «ИСТИНА».

● Чтение статуса программы, которая содержит команду PrgStatus

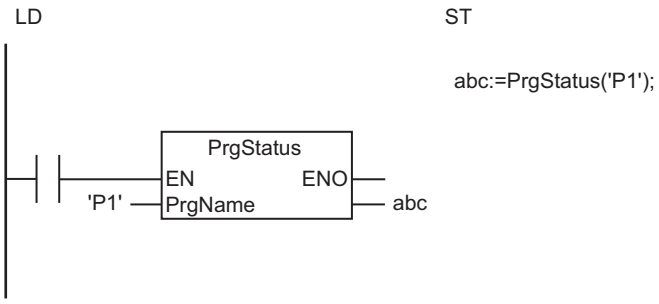
- В программе P1 цикла выполнения задачи 1 выполняется команда PrgStop, в которой указана программа P1.
- Затем в программе P1 цикла выполнения задачи 1 выполняется команда PrgStatus, в которой указана программа P1.
- Программа P1 была деактивирована для цикла выполнения задачи 2, поэтому выход *Out* команды PrgStatus содержит ЛОЖЬ.



P1 уже была деактивирована для цикла 2, когда выполнялась команда PrgStatus, поэтому выход ***Out*** команды PrgStatus содержит «ЛОЖЬ».

Пример записи

Ниже показан пример записи команды для чтения статуса программы с именем «P1».



Дополнительная информация

- Для активации указанной программы из программы пользователя используйте команду *PrgStart* на стр. 2-1022.
- Для деактивации указанной программы из программы пользователя используйте команду *PrgStop* на стр. 2-1032.

Меры предосторожности для обеспечения надлежащей эксплуатации

- В указанном ниже случае произойдет ошибка. Выход *Out* перейдет в состояние ЛОЖЬ.
  a) Программы с указанным именем *PrgName* не существует.

Пример программы

В данном примере имеются три программы: P1, P2 и P3. Смена выполняемой программы осуществляется с помощью сенсорной панели.

Описание сенсорной панели

В этом примере предполагается, что сенсорная панель подключена к контроллеру.
Ниже перечислены индикаторы, предусмотренные на сенсорной панели.

Наименование индикатора	Описание
Индикатор выполнения P1	Горит во время выполнения программы P1.
Индикатор выполнения P2	Горит во время выполнения программы P2.
Индикатор выполнения P3	Горит во время выполнения программы P3.

На сенсорной панели также предусмотрены указанные ниже кнопки.

Наименование кнопки	Действие при нажатии кнопки
Кнопка смены выполняемой программы	Каждое нажатие кнопки приводит к смене выполняемой программы в следующем порядке: P1 -> P2 -> P3, а затем снова P1.

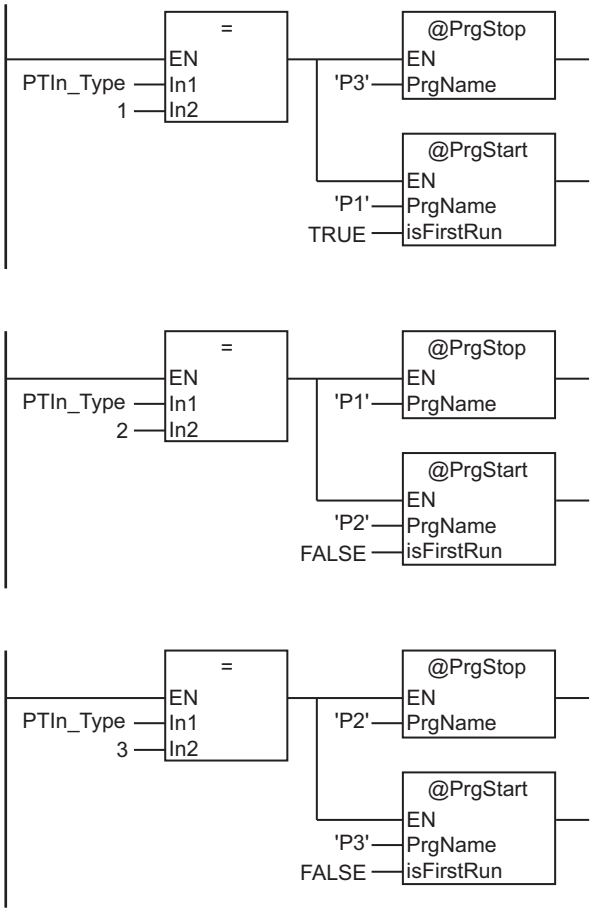
Глобальные переменные

Переменная	Тип данных	Начальное значение	Комментарий
PTIn_Type	INT	0	Вход кнопки смены выполняемой программы
PTOut_P1Status	BOOL	ЛОЖЬ	Выход индикатора выполнения P1
PTOut_P2Status	BOOL	ЛОЖЬ	Выход индикатора выполнения P2
PTOut_P3Status	BOOL	ЛОЖЬ	Выход индикатора выполнения P3

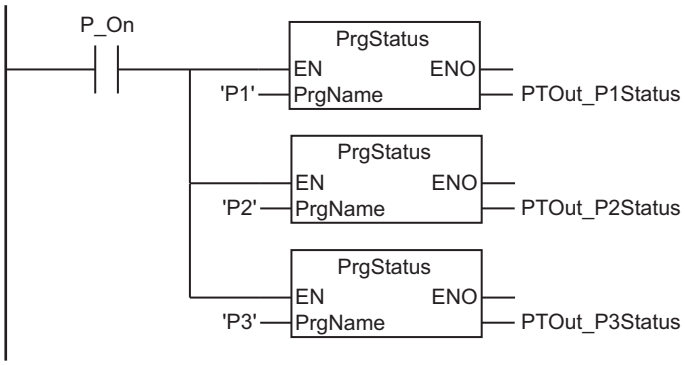
Программа на языке LD

Внешние переменные	Переменная	Тип данных	Комментарий
	PTIn_Type	INT	Вход кнопки смены выполняемой программы
	PTOut_P1Status	BOOL	Выход индикатора выполнения P1
	PTOut_P2Status	BOOL	Выход индикатора выполнения P2
	PTOut_P3Status	BOOL	Выход индикатора выполнения P3

Изменение выполняемой программы.



Выполнение команды PrgStatus.



Программа на языке ST

Внешние переменные	Переменная	Тип данных	Комментарий
	PTIn_Type	INT	Вход кнопки смены выполняемой программы
	PTOut_P1Status	BOOL	Выход индикатора выполнения P1
	PTOut_P2Status	BOOL	Выход индикатора выполнения P2
	PTOut_P3Status	BOOL	Выход индикатора выполнения P3

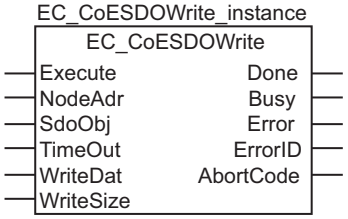
```
// Смена выполняемой программы.
```

```
IF PTIn_Type = 1 THEN
  PrgStop('P3');
  PrgStart('P1',TRUE);
ELSIF PTIn_Type = 2 THEN
  PrgStop('P1');
  PrgStart('P2',FALSE);
ELSIF PTIn_Type = 3 THEN
  PrgStop('P2');
  PrgStart('P3',FALSE);
END_IF;

// Выполнение команды PrgStatus.
IF P_On THEN
  PTOut_P1Status:=PrgStatus('P1');
  PTOut_P2Status:=PrgStatus('P2');
  PTOut_P3Status:=PrgStatus('P3');
END_IF;
```

Команды для обмена данными по интерфейсу EtherCAT

EC_CoESDOWrite

Команда EC_CoESDOWrite производит запись значения в объект CoE (CAN Application Protocol over EtherCAT) указанного ведомого устройства в сети EtherCAT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EC_CoESDO Write	Запись в SDO CoE в EtherCAT	FB		EC_CoESDOWrite_instance(Execute, NodeAdr, SdoObj, TimeOut, WriteDat, WriteSize, Done, Busy, Error, ErrorID, AbortCode);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
NodeAdr	Адрес узла ведомого устройства	Вход	Адрес узла ведомого устройства для доступа	1...512*1	---	---
SdoObj	Параметр SDO		Параметр SDO	---		
TimeOut	Время ожидания		0: 2,0 с 1...65535: 0,1...6553,5 с	Зависит от типа данных.	0,1 с	20 (2,0 с)
WriteDat	Записываемые данные		Записываемые данные		---	---
WriteSize	Объем записываемых данных		Объем записываемых данных*2	1…2048	Байты	
AbortCode	Код прекращения	Выход	Код ответа для доступа к SDO, указанный протоколом CoE 0: нормальное завершение	Зависит от типа данных.	---	---

*1. Допустимый диапазон для модулей ЦПУ NX102, NX1P2 и модуля ЦПУ серии NJ: *1...192*.

*2. Объем записываемых данных может быть меньше 1 байта, например при записи значения типа BOOL или массива типа BOOL. Если он меньше 1 байта, установите значение *WriteSize* равным 1.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NodeAdr							ОК													
SdoObj	Подробные сведения о структуре _sSDO_ACCESS см. в разделе *Функция* на стр. 2-1061.																			
TimeOut							ОК													

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
WriteDat	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать перечисление, массив, элемент массива, член структуры или член объединения.																			
WriteSize							OK													
AbortCode				OK																

Функция

Команда EC_CoESDOWrite производит запись данных в объект CoE узла, указанного параметром *NodeAdr* (адрес узла ведомого устройства).
В объект записывается содержимое переменной *WriteDat*. Объем записываемых данных указывается с помощью параметра *WriteSize*.
Параметр SDO задается с помощью параметра *SdoObj*.

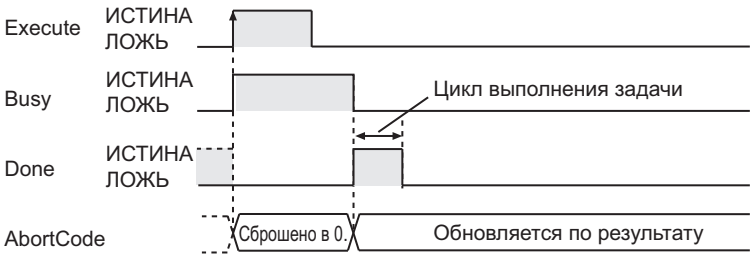
Для переменной *SdoObj* используется структурный тип данных _sSDO_ACCESS. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
SdoObj	Параметр SDO	Параметр SDO	_sSDO_ACCESS	---	---	---
Index	Индекс	Номер индекса в словаре объектов, определенном в CoE	UINT	1...65535	---	---
Subindex	Подындекс	Номер подындекса в словаре объектов, определенном в CoE	USINT	Зависит от типа данных.		
IsCompleteAccess	Полный доступ	Выбор полного доступа к SDO ИСТИНА: доступ к данным для всех подындексов ЛОЖЬ: доступ к данным для указанного подындекса	BOOL			

После завершения записи команда ожидает ответ в течение времени, заданного в *TimeOut*. Ответ хранится в переменной *AbortCode*.

При получении нормального ответа переменная *AbortCode* будет содержать 0. Значение в переменную *AbortCode* сохраняется, только если значение переменной *ErrorID* = 16#1804 (ответ с кодом прекращения SDO).
Значение и смысл кода прекращения в *AbortCode* зависит от ведомого устройства. См. руководство по соответствующему ведомому устройству.

Временная диаграмма, поясняющая сказанное, представлена на рисунке ниже. Значение в переменную *AbortCode* сохраняется после завершения выполнения команды, когда выход *Busy* переходит в состояние ЛОЖЬ.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_MBXSlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, способных участвовать в обмене сообщениями	BOOL	Эта переменная указывает возможность связи для каждого ведомого устройства. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.

Дополнительная информация

- Дополнительные сведения об интерфейсе связи EtherCAT см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя (Cat. No. W505)*.
- Коды прекращения SDO см. в разделе *A-5 Коды прерывания SDO* на стр. A-39.

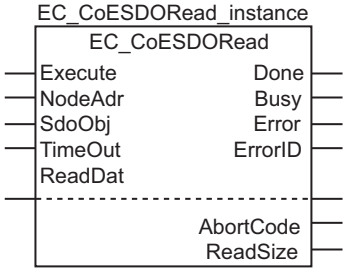
Меры предосторожности для обеспечения надлежащей эксплуатации

- Всегда используйте переменную в качестве входного параметра, передаваемого в *WriteDat*. При передаче константы произойдет ошибка сборки.
- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать только для портов EtherCAT Серия NJ/NX.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Ведущее устройство EtherCAT не находится в состоянии, в котором разрешен обмен сообщениями.
  b) Ведомого устройства с указанным адресом узла *NodeAdr* не существует.
  c) Ведомое устройство с указанным адресом узла *NodeAdr* не находится в состоянии, в котором разрешен обмен данными.
  d) Ведомое устройство возвращает ответ с ошибкой.

e) Было выполнено более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.

EC_CoESDORead

Команда EC_CoESDORead производит чтение значения из объекта CoE (CAN Application Protocol over EtherCAT) указанного ведомого устройства в сети EtherCAT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EC_CoESDO Read	Чтение из SDO CoE в EtherCAT	FB		EC_CoESDORead_instance(Execute, NodeAdr, SdoObj, TimeOut, ReadDat, Done, Busy, Error, ErrorID, AbortCode, ReadSize);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
NodeAdr	Адрес узла ведомого устройства	Вход	Адрес узла ведомого устройства для доступа	1...512*1	---	---
SdoObj	Параметр SDO		Параметр SDO	---		
TimeOut	Время ожидания		0: 2,0 с 1...65535: 0,1...6553,5 с	Зависит от типа данных.	0,1 с	0 (2,0 с)
AbortCode	Код прекращения	Выход	Код ответа для доступа к SDO, указанный протоколом CoE 0: нормальное завершение	Зависит от типа данных.	---	---
ReadSize	Объем прочитанных данных		Объем данных, сохраненных в *ReadDat*, после завершения чтения данных*2		Байты	
ReadDat	Прочитанные данные	Вход-выход	Буфер прочитанных данных	Зависит от типа данных.	---	---

*1. Допустимый диапазон для модулей ЦПУ NX102, NX1P2 и модуля ЦПУ серии NJ: *1...192*.

*2. Объем прочитанных данных может быть меньше 1 байта, например, если читается значение типа BOOL или массив типа BOOL. Если он меньше 1 байта, установите значение *ReadSize* равным 1.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NodeAdr							OK													

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SdoObj	Подробные сведения о структуре _sSDO_ACCESS см. в разделе *Функция* на стр. 2-1065.																			
TimeOut							OK													
AbortCode				OK																
ReadSize							OK													
ReadDat	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать перечисление, массив, элемент массива, член структуры или член объединения.																			

Функция

Команда EC_CoESDORead производит чтение данных из объекта CoE узла, указанного параметром *NodeAdr* (адрес узла ведомого устройства).
Прочитанные данные сохраняются в переменную *ReadDat*. Объем сохраненных данных записывается в переменную *ReadSize*. Значение в переменной *ReadSize* действительно, только если данные были успешно сохранены.
Параметр SDO задается с помощью параметра *SdoObj*.

Для переменной *SdoObj* используется структурный тип данных _sSDO_ACCESS. Описание приведено в таблице ниже.

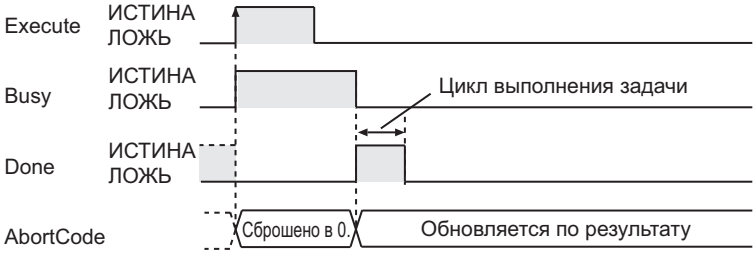
Имя		Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
SdoObj		Параметр SDO	Параметр SDO	_sSDO_ACCESS	---	---	---
	Index	Индекс	Номер индекса в словаре объектов, определенном в CoE	UINT	1…65535	---	---
	Subindex	Подындекс	Номер подындекса в словаре объектов, определенном в CoE	USINT	Зависит от типа данных.		
	IsCompleteAccess	Полный доступ	Выбор полного доступа к SDO ИСТИНА: доступ к данным для всех подындексов ЛОЖЬ: доступ к данным для указанного подындекса	BOOL			

После завершения чтения команда ожидает ответ в течение времени, заданного в *TimeOut*. Ответ хранится в переменной *AbortCode*.

При получении нормального ответа переменная *AbortCode* будет содержать 0. Значение в переменную *AbortCode* сохраняется, только если значение переменной *ErrorID* = 16#1804 (ответ с кодом прекращения SDO).

Значение и смысл кода прекращения в *AbortCode* зависит от ведомого устройства. См. руководство по соответствующему ведомому устройству.

Временная диаграмма, поясняющая сказанное, представлена на рисунке ниже. Значение в переменную *AbortCode* сохраняется после завершения выполнения команды, когда выход *Busy* переходит в состояние ЛОЖЬ.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_MBXSlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, способных участвовать в обмене сообщениями	BOOL	Эта переменная указывает возможность связи для каждого ведомого устройства. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.

Дополнительная информация

- Дополнительные сведения об интерфейсе связи EtherCAT см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя (Cat. No. W505)*.
- Коды прекращения SDO см. в разделе *A-5 Коды прерывания SDO* на стр. A-39.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать только для портов EtherCAT Серия NJ/NX.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки.
  a) Ведущее устройство EtherCAT не находится в состоянии, в котором разрешен обмен сообщениями.
  b) Ведомого устройства с указанным адресом узла *NodeAdr* не существует.
  c) Ведомое устройство с указанным адресом узла *NodeAdr* не находится в состоянии, в котором разрешен обмен данными.

d) Ведомое устройство возвращает ответ с ошибкой.
e) Объем прочитанных данных больше размера *ReadDat*.
f) Было выполнено более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.

Пример программы

В этом примере с помощью сообщения SDO EtherCAT будет прочитана версия программного обеспечения сервопривода OMRON серии 1S. Адрес узла ведомого устройства: 1.

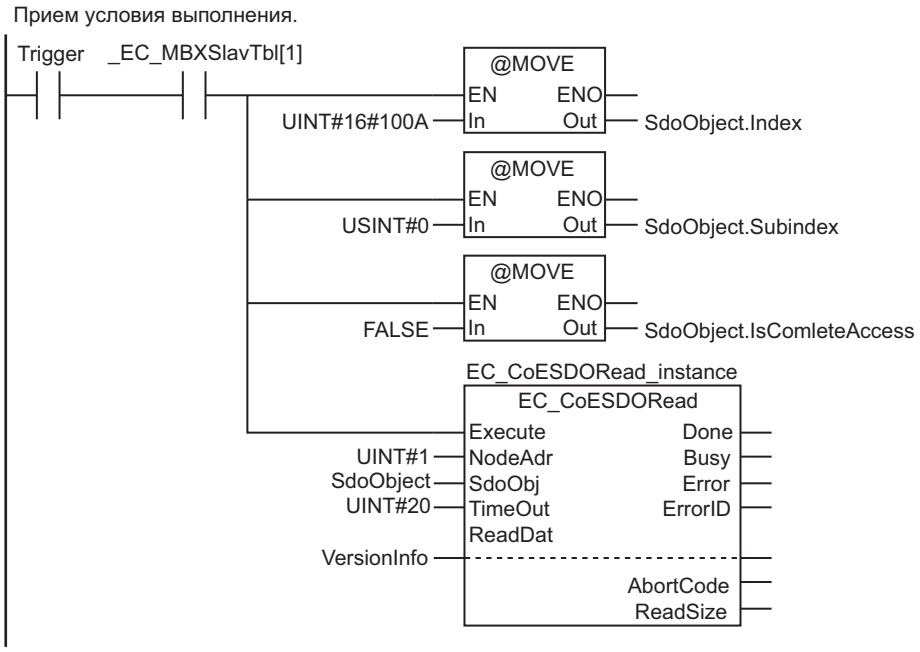
Индекс объекта с информацией о версии программного обеспечения: 16#100A. Подындекс: 0. Прочитанное значение сохраняется в переменную *VersionInfo* типа STRING.

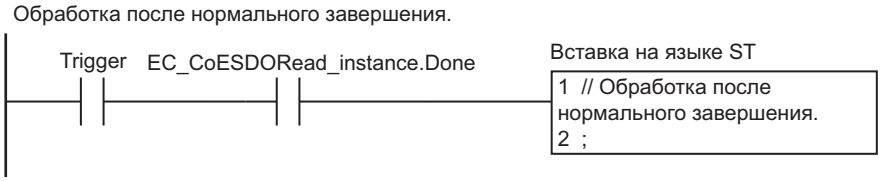
Программа на языке LD

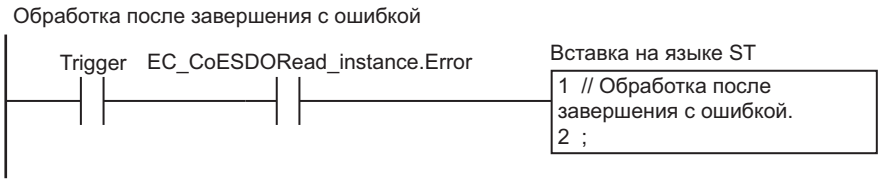
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	SdoObject	_sSDO_ACCESS	(Index:=0, Subindex:=0, IsCompleteAccess:=FALSE)	Параметр SDO
	VersionInfo	STRING[256]	''	Прочитанные данные
	EC_CoESDORead_instance	EC_CoESDORead		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EC_MBXSlavTbl	ARRAY[1..512] OF BOOL*1	☑	Таблица ведомых устройств, способных участвовать в обмене сообщениями

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.







Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	SdoObject	_sSDO_ACCESS	(Index:=0, Subindex:=0, IsCompleteAccess:=FALSE)	Параметр SDO
	DoSdoRead	BOOL	ЛОЖЬ	Обработка
	VersionInfo	STRING[256]	''	Прочитанные данные
	NormalEnd	UINT	0	Нормальное завершение
	ErrorEnd	UINT	0	Завершение с ошибкой
	EC_CoESDORead_instance	EC_CoESDORead		

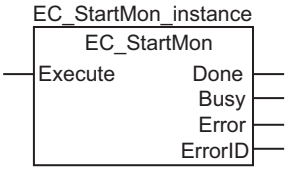
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EC_MBXSlavTbl	ARRAY[1..512] OF BOOL*1	☑	Таблица ведомых устройств, способных участвовать в обмене сообщениями

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

```
// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (DoSdoRead=FALSE) AND (_EC_MBXSlavTbl[1]=TRUE) ) THEN
  DoSdoRead :=TRUE;
  SdoObject.Index :=UINT#16#100A;
  SdoObject.Subindex :=USINT#0;
  SdoObject.IsCompleteAccess:=FALSE;
  EC_CoESDORead_instance(
    Execute:=FALSE, // Инициализация экземпляра.
    ReadDat:=VersionInfo); // Пустышка
END_IF;

// Выполнение команды EC_CoESDORead.
IF (DoSdoRead=TRUE) THEN
  EC_CoESDORead_instance(
    Execute:=TRUE,
    NodeAdr:=UINT#1, // Адрес узла 1
    SdoObj :=SdoObject, // Параметр SDO
    TimeOut:=UINT#20, // Время ожидания: 2,0 с
    ReadDat:=VersionInfo); // Прочитанные данные

  IF (EC_CoESDORead_instance.Done=TRUE) THEN
    // Обработка после нормального завершения
    NormalEnd:=NormalEnd+UINT#1;
  ELSIF (EC_CoESDORead_instance.Error=TRUE) THEN
    // Обработка после завершения с ошибкой
    ErrorEnd :=ErrorEnd+UINT#1;
  END_IF;

END_IF;
```

EC_StartMon

Команда EC_StartMon запускает мониторинг пакетов для интерфейса связи EtherCAT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EC_StartMon	Запуск мониторинга пакетов EtherCAT	FB		EC_StartMon_instance(Execute, Done, Busy, Error, ErrorID);

Информация о версии

В зависимости от версии модуля ЦПУ и версии Sysmac Studio могут действовать следующие ограничения:

- Эту команду нельзя использовать, если в проекте указана версия модуля 1.40 или более поздняя.
- В случае модулей ЦПУ NX701 и NJ101 команду можно использовать с Sysmac Studio версии 1.13 или выше.
- В случае модуля ЦПУ NX1P2 команду можно использовать с Sysmac Studio версии 1.17 или выше.
- В случае модуля ЦПУ NJ301 команду можно использовать с версией модуля 1.10 или более поздней и Sysmac Studio версии 1.12 или выше.

Переменные

Используются только общие переменные.

Функция

Команда EC_StartMon запускает выполнение мониторинга пакетов для интерфейса связи EtherCAT.

Функция мониторинга пакетов собирает указанное количество самых последних коммуникационных пакетов EtherCAT.

После достижения указанного количества пакетов старые пакеты отбрасываются по порядку.

После выполнения команды EC_StartMon мониторинг пакетов продолжается до тех пор, пока не будет выполнена команда EC_StopMon.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_PktMonStop	Мониторинг пакетов остановлен	BOOL	Эта переменная показывает, остановлен ли мониторинг пакетов. ИСТИНА: остановлен. ЛОЖЬ: не остановлен.

Имя	Значение	Тип данных	Описание
_EC_PktSaving	Сохранение файла данных пакетов	BOOL	Эта переменная показывает, сохраняет ли в данный момент команда данные пакетов во внутренний файл в оперативной памяти модуля ЦПУ. ИСТИНА: сохраняет. ЛОЖЬ: не сохраняет.

Дополнительная информация

- Собранные данные пакетов не могут быть сохранены во внутренний файл в оперативной памяти модуля ЦПУ во время выполнения команды ECATStartMonitor.
- Для сохранения данных пакетов во внутренний файл в оперативной памяти модуля ЦПУ необходимо выполнить команду EC_StopMon, чтобы остановить мониторинг пакетов, после чего нужно выполнить команду EC_SaveMon, чтобы сохранить пакеты.
- Дополнительные сведения об интерфейсе связи EtherCAT см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя (Cat. No. W505)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать только для портов EtherCAT Серия NJ/NX.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки.
  a) В проекте используется версия модуля 1.40 или более поздняя.
  b) В данный момент выполняется операция сохранения данных пакетов во внутренний файл в оперативной памяти модуля ЦПУ.
  c) Было выполнено более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.

Пример программы

Рассмотрим в качестве примера программу, которая передает коммуникационные пакеты EtherCAT на карту памяти SD при возникновении ошибки ведомого устройства EtherCAT. Используется файл с именем «PacketFile».
Обработка выполняется в следующем порядке:

1 Контролируется системная переменная *_EC_ErrSta* (Ошибка EtherCAT), и при возникновении ошибки запускается процедура передачи пакетов в файл.

2 Используется команда EC_StopMon для остановки мониторинга пакетов для интерфейса связи EtherCAT.

3 С помощью команды EC_SaveMon данные коммуникационных пакетов EtherCAT сохраняются во внутренний файл в оперативной памяти модуля ЦПУ.

4 С помощью команды EC_CopyMon этот файл копируется на карту памяти SD.

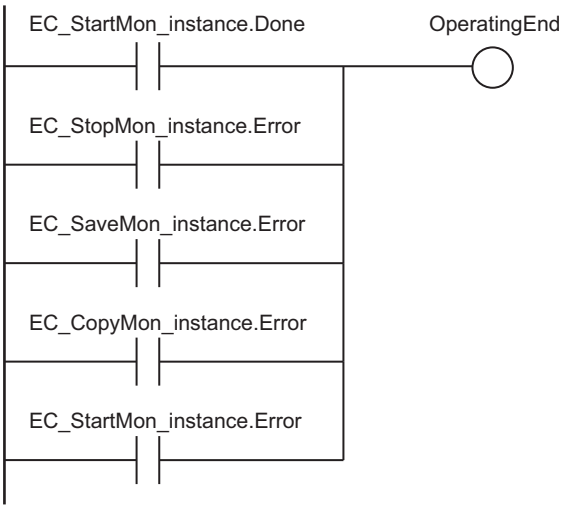
5 Используется команда EC_StartMon для повторного запуска мониторинга пакетов для интерфейса связи EtherCAT.

Программа на языке LD

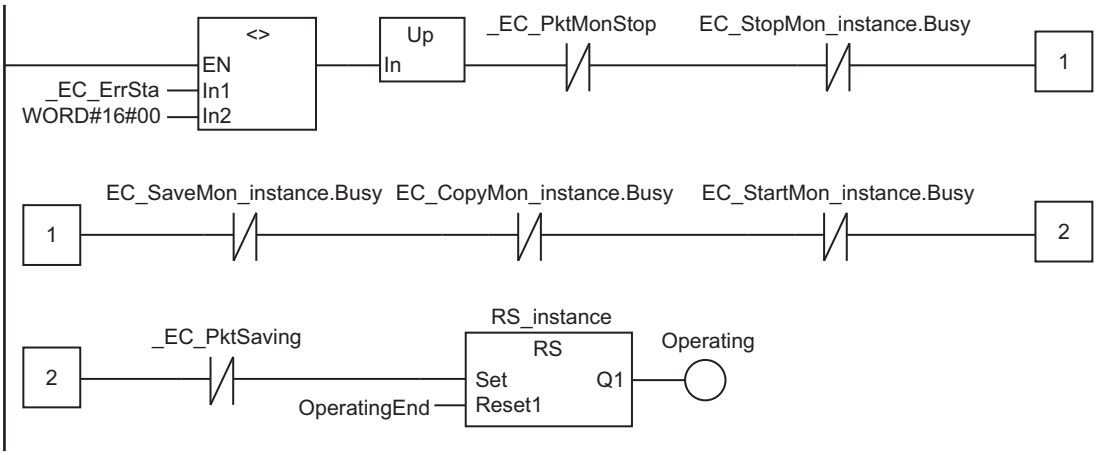
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена
	Operating	BOOL	ЛОЖЬ	Условие выполнения
	RS_instance	RS		
	EC_StopMon_instance	EC_StopMon		
	EC_SaveMon_instance	EC_SaveMon		
	EC_CopyMon_instance	EC_CopyMon		
	EC_StartMon_instance	EC_StartMon		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EC_ErrSta	WORD	☑	Ошибка встроенного интерфейса EtherCAT
	_EC_PktMonStop	BOOL	☑	Мониторинг пакетов остановлен
	_EC_PktSaving	BOOL	☑	Сохранение файла данных пакетов
	_Card1Ready	BOOL	☑	Флаг готовности карты памяти SD

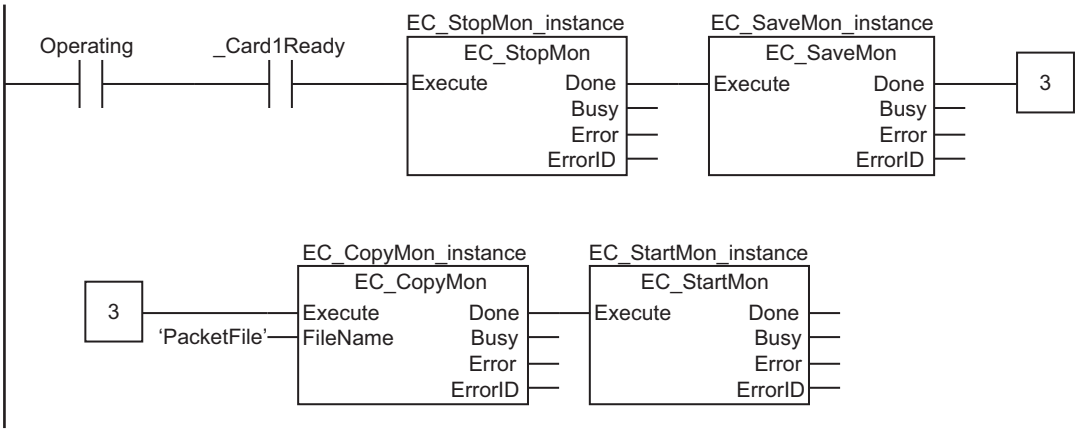
Определение, завершено ли выполнение команды.



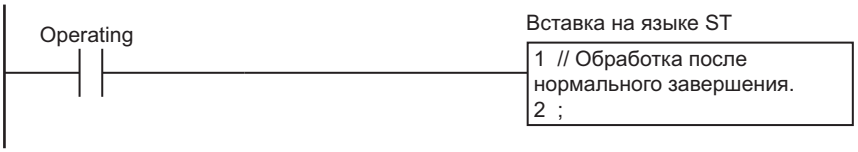
Мониторинг ошибок сети EtherCAT.



Выполнение команды



Обработка после нормального завершения.



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	EC_Err	BOOL	ЛОЖЬ	Ошибка контроллера в функциональном модуле «EtherCAT Master».
	EC_Err_Trigger	BOOL	ЛОЖЬ	Определение перехода EC_Err в состояние ИСТИНА.
	DoEC_PktSave	BOOL	ЛОЖЬ	Обработка
	Stage	INT	0	Смена этапа
	R_TRIG_instance	R_TRIG		
	EC_StopMon_instance	EC_StopMon		
	EC_SaveMon_instance	EC_SaveMon		
	EC_CopyMon_instance	EC_CopyMon		
	EC_StartMon_instance	EC_StartMon		

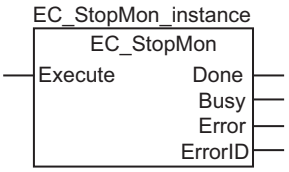
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EC_ErrSta	WORD	☑	Ошибка встроенного интерфейса EtherCAT
	_EC_PktMonStop	BOOL	☑	Мониторинг пакетов остановлен
	_EC_PktSaving	BOOL	☑	Сохранение файла данных пакетов
	_Card1Ready	BOOL	☑	Флаг готовности карты памяти SD

```
// Запуск последовательности при переходе _EC_ErrSta в состояние ИСТИНА.
EC_Err:=(_EC_ErrSta <> WORD#16#00);
R_TRIG_instance(Clk:=EC_Err, Q=>EC_Err_Trigger);

IF ( (EC_Err_Trigger=TRUE) AND (DoEC_PktSave=FALSE) AND (_EC_PktMonStop=FALSE)
  AND (_EC_PktSaving=FALSE) AND (_Card1Ready=TRUE) ) THEN
  DoEC_PktSave:=TRUE;
  Stage :=INT#1;
  EC_StopMon_instance(Execute:=FALSE); // Инициализация экземпляра.
  EC_SaveMon_instance(Execute:=FALSE);
  EC_CopyMon_instance(Execute:=FALSE);
  EC_StartMon_instance(Execute:=FALSE);
END_IF;

// Выполнение команды
IF (DoEC_PktSave=TRUE) THEN
  CASE Stage OF
  1 : // Остановка мониторинга пакетов EtherCAT.
    EC_StopMon_instance(
      Execute:=TRUE);
```

```
    IF (EC_StopMon_instance.Done=TRUE) THEN
      Stage:=INT#2; // Нормальное завершение
    ELSIF (EC_StopMon_instance.Error=TRUE) THEN
      Stage:=INT#10; // Завершение с ошибкой
    END_IF;

  2 : // Сохранение данных пакетов EtherCAT во внутренний файл.
    EC_SaveMon_instance(
      Execute:=TRUE);
    IF (EC_SaveMon_instance.Done=TRUE) THEN
      Stage:=INT#3; // Нормальное завершение
    ELSIF (EC_SaveMon_instance.Error=TRUE) THEN
      Stage:=INT#20; // Завершение с ошибкой
    END_IF;

  3 : // Копирование файла данных пакетов EtherCAT на карту памяти SD.
    EC_CopyMon_instance(
      Execute :=TRUE,
      FileName:='PacketFile');
    IF (EC_CopyMon_instance.Done=TRUE) THEN
      Stage:=INT#4; // Нормальное завершение
    ELSIF (EC_CopyMon_instance.Error=TRUE) THEN
      Stage:=INT#30; // Завершение с ошибкой
    END_IF;

  4 : // Перезапуск мониторинга пакетов EtherCAT.
    EC_StartMon_instance(
      Execute:=TRUE);
    IF (EC_StartMon_instance.Done=TRUE) THEN
      Stage:=INT#0; // Нормальное завершение
    ELSIF (EC_StartMon_instance.Error=TRUE) THEN
      Stage:=INT#40; // Завершение с ошибкой
    END_IF;

  0 : // Обработка после нормального завершения
    DoEC_PktSave:=FALSE;

  ELSE // Обработка после завершения с ошибкой
    DoEC_PktSave:=FALSE;
  END_CASE;
END_IF;
```

EC_StopMon

Команда EC_StopMon останавливает выполнение мониторинга пакетов для интерфейса связи EtherCAT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EC_StopMon	Остановка мониторинга пакетов EtherCAT	FB		EC_StopMon_instance(Execute, Done, Busy, Error, ErrorID);

Информация о версии

В зависимости от версии модуля ЦПУ и версии Sysmac Studio могут действовать следующие ограничения:

- Эту команду нельзя использовать, если в проекте указана версия модуля 1.40 или более поздняя.
- В случае модулей ЦПУ NX701 и NJ101 команду можно использовать с Sysmac Studio версии 1.13 или выше.
- В случае модуля ЦПУ NX1P2 команду можно использовать с Sysmac Studio версии 1.17 или выше.
- В случае модуля ЦПУ NJ301 команду можно использовать с версией модуля 1.10 или более поздней и Sysmac Studio версии 1.12 или выше.

Переменные

Используются только общие переменные.

Функция

Команда EC_StopMon останавливает выполнение мониторинга пакетов для интерфейса связи EtherCAT.

Функция мониторинга пакетов собирает указанное количество самых последних коммуникационных пакетов EtherCAT.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_PktMonStop	Мониторинг пакетов остановлен	BOOL	Эта переменная показывает, остановлен ли мониторинг пакетов. ИСТИНА: остановлен. ЛОЖЬ: не остановлен.
_EC_PktSaving	Сохранение файла данных пакетов	BOOL	Эта переменная показывает, сохраняет ли в данный момент команда данные пакетов во внутренний файл в оперативной памяти модуля ЦПУ. ИСТИНА: сохраняет. ЛОЖЬ: не сохраняет.

Дополнительная информация

- При сохранении собранных данных пакетов во внутренний файл в оперативной памяти модуля ЦПУ необходимо выполнить эту команду, чтобы остановить работу функции мониторинга пакетов, после чего нужно выполнить команду EC_SaveMon, чтобы сохранить данные.
- Дополнительные сведения об интерфейсе связи EtherCAT см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя (Cat. No. W505)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

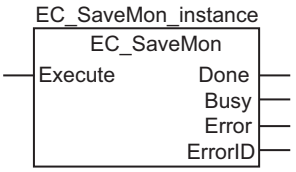
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать только для портов EtherCAT Серия NJ/NX.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки.
  a) В проекте используется версия модуля 1.40 или более поздняя.
  b) Мониторинг пакетов уже остановлен.
  c) Было выполнено более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.

Пример программы

См. *Пример программы* на стр. 2-1071 для команды EC_StartMon.

EC_SaveMon

Команда EC_SaveMon сохраняет данные коммуникационных пакетов EtherCAT во внутренний файл в оперативной памяти модуля ЦПУ.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EC_SaveMon	Сохранение пакетов EtherCAT	FB		EC_SaveMon_instance(Execute, Done, Busy, Error, ErrorID);

Информация о версии

В зависимости от версии модуля ЦПУ и версии Sysmac Studio могут действовать следующие ограничения:

- Эту команду нельзя использовать, если в проекте указана версия модуля 1.40 или более поздняя.
- В случае модулей ЦПУ NX701 и NJ101 команду можно использовать с Sysmac Studio версии 1.13 или выше.
- В случае модуля ЦПУ NX1P2 команду можно использовать с Sysmac Studio версии 1.17 или выше.
- В случае модуля ЦПУ NJ301 команду можно использовать с версией модуля 1.10 или более поздней и Sysmac Studio версии 1.12 или выше.

Переменные

Используются только общие переменные.

Функция

Команда EC_SaveMon сохраняет данные коммуникационных пакетов EtherCAT, которые были собраны функцией мониторинга пакетов, во внутренний файл в оперативной памяти модуля ЦПУ.

Функция мониторинга пакетов собирает указанное количество самых последних коммуникационных пакетов EtherCAT.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_PktMonStop	Мониторинг пакетов остановлен	BOOL	Эта переменная показывает, остановлен ли мониторинг пакетов. ИСТИНА: остановлен. ЛОЖЬ: не остановлен.
_EC_PktSaving	Сохранение файла данных пакетов	BOOL	Эта переменная показывает, сохраняет ли в данный момент команда данные пакетов во внутренний файл в оперативной памяти модуля ЦПУ. ИСТИНА: сохраняет. ЛОЖЬ: не сохраняет.

Дополнительная информация

- Во время выполнения этой команды невозможно выполнять мониторинг пакетов.
- Дополнительные сведения об интерфейсе связи EtherCAT см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя (Cat. No. W505)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

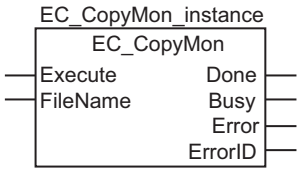
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать только для портов EtherCAT Серия NJ/NX.
- Эту команду невозможно выполнить во время выполнения мониторинга пакетов. Сначала следует выполнить команду EC_StopMon, чтобы остановить мониторинг пакетов.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки.
  a) В проекте используется версия модуля 1.40 или более поздняя.
  b) В данный момент выполняется мониторинг пакетов.
  c) Выполняется более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.

Пример программы

См. *Пример программы* на стр. 2-1071 для команды EC_StartMon.

EC_CopyMon

Команда EC_CopyMon передает данные пакетов во внутреннем файле в оперативной памяти модуля ЦПУ на карту памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EC_CopyMon	Передача пакетов EtherCAT	FB		EC_CopyMon_instance(Execute, FileName, Done, Busy, Error, ErrorID);

Информация о версии

В зависимости от версии модуля ЦПУ и версии Sysmac Studio могут действовать следующие ограничения:

- Эту команду нельзя использовать, если в проекте указана версия модуля 1.40 или более поздняя.
- В случае модулей ЦПУ NX701 и NJ101 команду можно использовать с Sysmac Studio версии 1.13 или выше.
- В случае модуля ЦПУ NX1P2 команду можно использовать с Sysmac Studio версии 1.17 или выше.
- В случае модуля ЦПУ NJ301 команду можно использовать с версией модуля 1.10 или более поздней и Sysmac Studio версии 1.12 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileName	Имя файла	Вход	Имя файла на карте памяти SD	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileName																				OK

Функция

Команда EC_CopyMon передает данные пакетов, содержащиеся во внутреннем файле в оперативной памяти модуля ЦПУ, на карту памяти SD.
Имя файла на карте памяти SD указывается параметром *FileName*.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_PktSaving	Сохранение файла данных пакетов	BOOL	Эта переменная показывает, сохраняет ли в данный момент команда данные пакетов во внутренний файл в оперативной памяти модуля ЦПУ. ИСТИНА: сохраняет. ЛОЖЬ: не сохраняет.

Дополнительная информация

Дополнительные сведения об интерфейсе связи EtherCAT см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя (Cat. No. W505)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

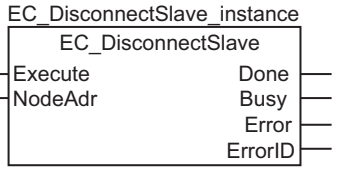
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать только для портов EtherCAT Серия NJ/NX.
- Эту команду невозможно выполнить во время выполнения операции сохранения пакетов.
- Для использования данной команды сначала необходимо выполнить команду EC_SaveMon, чтобы сохранить данные пакетов во внутренний файл в оперативной памяти модуля ЦПУ.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки.
  a) В проекте используется версия модуля 1.40 или более поздняя.
  b) В данный момент выполняется операция сохранения файла данных пакетов.
  c) Было выполнено более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.

Пример программы

См. *Пример программы* на стр. 2-1071 для команды EC_StartMon.

EC_DisconnectSlave

Команда EC_DisconnectSlave отсоединяет указанное ведомое устройство от сети EtherCAT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EC_DisconnectSlave	Отсоединение ведомого устройства EtherCAT	FB		EC_DisconnectSlave_instance(Execute, NodeAdr, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
NodeAdr	Адрес узла ведомого устройства	Вход	Адрес узла отсоединяемого ведомого устройства	1...512*1	---	---

*1. Допустимый диапазон для модулей ЦПУ NX102, NX1P2 и модуля ЦПУ серии NJ: *1...192*.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NodeAdr							ОК													

Функция

Команда EC_DisconnectSlave отсоединяет от сети EtherCAT ведомое устройство, указанное параметром *NodeAdr* (адрес узла ведомого устройства).
В данном случае под отсоединением от сети подразумевается, что ведомое устройство переводится в состояние, в котором оно не работает, хотя по-прежнему присутствует в сети.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_EntrySlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, подключенных к сети	BOOL[]	Эта переменная показывает, являются ли ведомые устройства частью сети (т. е. присутствуют ли они в ней). ИСТИНА: является частью сети. ЛОЖЬ: не является частью сети.

Имя	Значение	Тип данных	Описание
_EC_DisconnSlavTbl[i] "i" — это адрес узла.	Таблица отсоединенных ведомых устройств	BOOL[]	Эта переменная показывает ведомые устройства, для которых в настоящее время действуют команды отсоединения от сети. ИСТИНА: действует команда отсоединения. ЛОЖЬ: не действует команда отсоединения.
_EC_DisableSlavTbl[i] "i" — это адрес узла.	Таблица деактивированных ведомых устройств	BOOL[]	Эта переменная показывает, не деактивировано ли то или иное ведомое устройство в сети. ИСТИНА: деактивировано. ЛОЖЬ: не деактивировано или не является частью сети.

Дополнительная информация

Дополнительные сведения об интерфейсе связи EtherCAT см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя (Cat. No. W505)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать только для портов EtherCAT Серия NJ/NX.
- Если к выходному порту отсоединяемого ведомого устройства (цепочечное подключение) подключены другие ведомые устройства, они также будут отсоединены от сети EtherCAT.
- Эту команду невозможно выполнить во время выполнения следующих команд: EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, ResetECError, RestartNXUnit и NX_ChangeWriteMode.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
- Эту команду нельзя использовать для отсоединения ведомых устройств в сети с кольцевой топологией. В то же время, ее можно использовать для отсоединения ведомых устройств, находящихся в ответвлении от кольца.
- В указанном ниже случае происходит ошибка. Значение *Error* поменяется на ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки.
  a) Ведомое устройство с указанным адресом узла *NodeAdr* не является частью сети EtherCAT. То есть элемент _EC_EntrySlavTbl[i] (Таблица ведомых устройств, подключенных к сети) для этого устройства содержит значение ЛОЖЬ.
  b) Ведомое устройство с указанным адресом узла *NodeAdr* деактивировано.
  c) Уже выполняется команда EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, ResetECError, RestartNXUnit или NX_ChangeWriteMode.
  d) Было выполнено более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.

Пример программы

В данном примере ведомое устройство 1 отсоединяется от сети EtherCAT, а затем снова подсоединяется к ней.
Когда значение переменной *Trigger1* меняется на ИСТИНА, выполняется команда EC_DisconnectSlave для отсоединения ведомого устройства 1. Когда значение переменной *Trigger2* меняется на ИСТИНА, выполняется команда EC_ConnectSlave для подсоединения ведомого устройства 1.

Исключение одновременного выполнения команд

Команды EC_DisconnectSlave и EC_ConnectSlave не допускается выполнять одновременно. Обе эти команды выполняются дольше одного периода выполнения задачи.
Прежде чем выполнять новую команду, необходимо убедиться в том, что завершилось выполнение предыдущей команды.
Для этой цели используется переменная *ExclusiveFlg* (Флаг исключения одновременного выполнения команд).
Если флаг *ExclusiveFlg* = ИСТИНА, значит одна из команд в данный момент выполняется.
Пока значение флага *ExclusiveFlg* = ИСТИНА, следующую команду выполнять не следует.
Команды EC_DisconnectSlave и EC_ConnectSlave не допускается выполнять одновременно не только в пределах одной задачи, но и в разных задачах.
Поэтому в рассматриваемом примере флаг *ExclusiveFlg* определен как глобальная переменная. Благодаря этому в программе можно реализовать эксклюзивное управление, чтобы исключить одновременное выполнение этих команд в других задачах. Эта же глобальная переменная *ExclusiveFlg* должна использоваться и в других задачах для той же цели (т. е. для исключения одновременного выполнения команд).
Команду EC_ChangeEnableSetting нельзя выполнять одновременно с командой EC_DisconnectSlave или EC_ConnectSlave.
Эта же глобальная переменная *ExclusiveFlg* используется в разделе *Пример программы* на стр. 2-1096 для команды EC_ChangeEnable для пояснения работы эксклюзивного управления командами.

Определения глобальных переменных

● Глобальные переменные

Переменная	Тип данных	Начальное значение	Комментарий
ExclusiveFlg	BOOL	ЛОЖЬ	Флаг исключения одновременного выполнения команд

Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Operating1End	BOOL	ЛОЖЬ	Обработка 1 завершена.

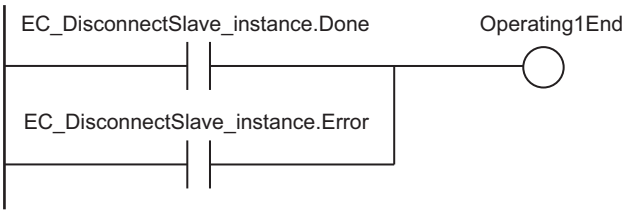
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger1	BOOL	ЛОЖЬ	Условие выполнения 1
	Operating1	BOOL	ЛОЖЬ	Обработка 1
	RS_instance1	RS		
	EC_DisconnectSlave_instance	EC_DisconnectSlave		
	Operating2End	BOOL	ЛОЖЬ	Обработка 2 завершена.
	Trigger2	BOOL	ЛОЖЬ	Условие выполнения 2
	Operating2	BOOL	ЛОЖЬ	Обработка 2
	RS_instance2	RS		
	EC_ConnectSlave_instance	EC_ConnectSlave		
	HMI_ConnectErrorID*1	WORD	16#0000	

*1. Переменные, имя которых начинается с *HMI_*, являются переменными для отображения на сенсорной панели.

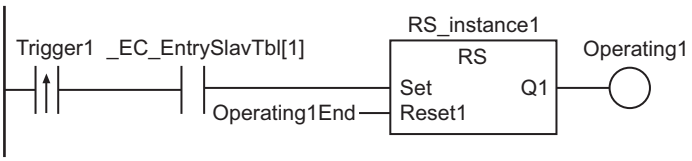
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EC_EntrySlavTbl	ARRAY[1..512] OF BOOL*1	☑	Таблица ведомых устройств, подключенных к сети
	_EC_DisconnSlavTbl	ARRAY[1..512] OF BOOL*1	☑	Таблица отсоединенных ведомых устройств
	ExclusiveFlg	BOOL	☐	Флаг исключения одновременного выполнения команд

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

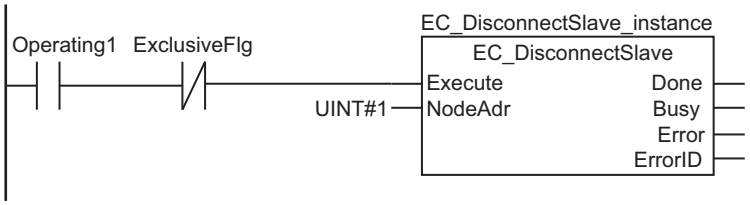
Проверка завершения выполнения команды EC_DisconnectSlave.



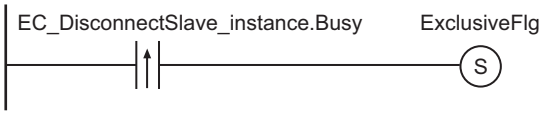
Прием условия выполнения 1.

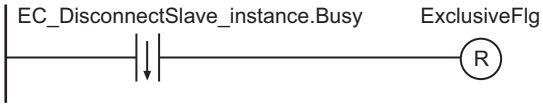


Выполнение команды EC_DisconnectSlave.

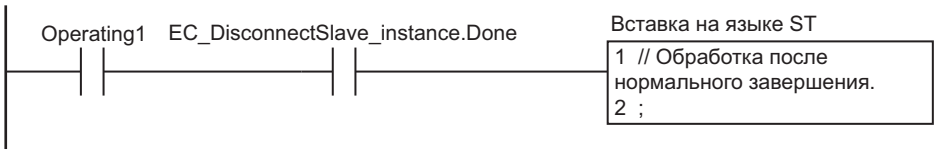


Исключение одновременного выполнения команд

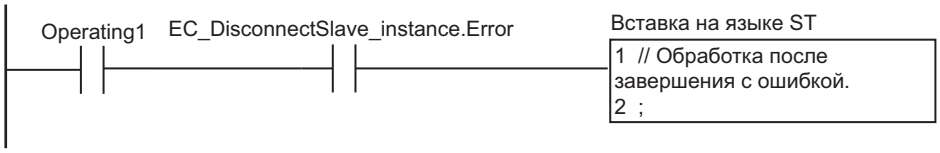




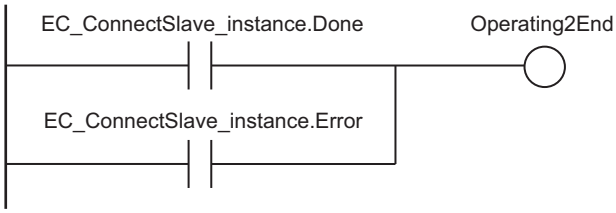
Обработка после нормального завершения.



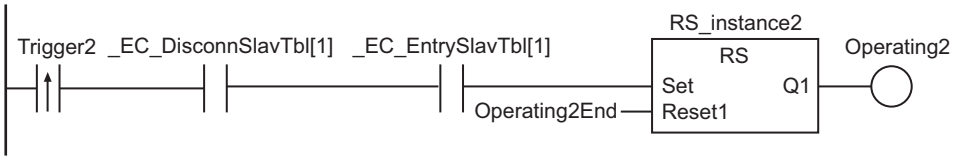
Обработка после завершения с ошибкой



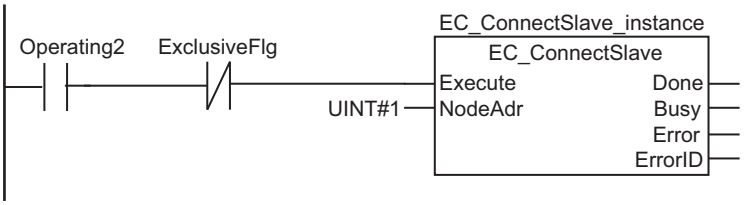
Проверка завершения выполнения команды EC_ConnectSlave.



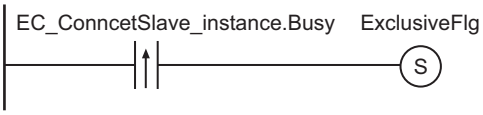
Прием условия выполнения 2.

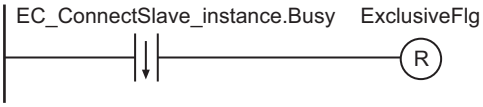


Выполнение команды EC_ConnectSlave.

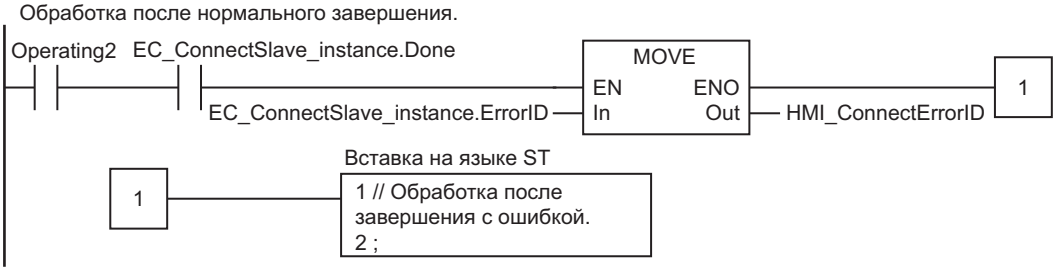


Исключение одновременного выполнения команд

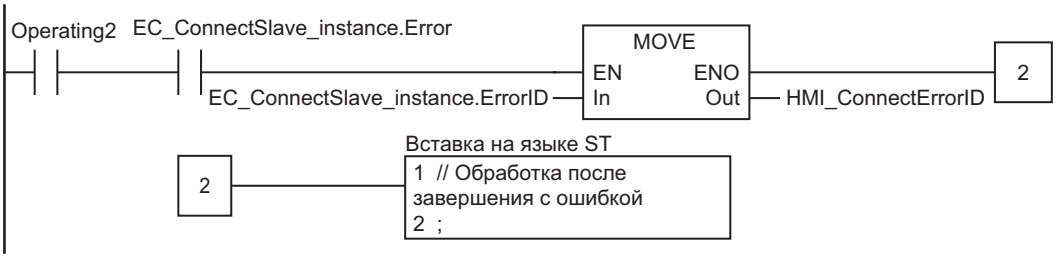




Обработка после нормального завершения.



Обработка после завершения с ошибкой



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger1	BOOL	ЛОЖЬ	Условие выполнения 1
	LastTrigger1	BOOL	ЛОЖЬ	Значение *Trigger1* в предыдущем цикле выполнения задачи
	Operating1Start	BOOL	ЛОЖЬ	Обработка 1 началась.
	Operating1	BOOL	ЛОЖЬ	Обработка 1
	DisconnectSet	BOOL	ЛОЖЬ	Идет выполнение команды EC_DisconnectSlave.
	DisconnectReset	BOOL	ЛОЖЬ	Команда EC_DisconnectSlave в данный момент не выполняется.

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	EC_DisconnectSlave_instance	EC_DisconnectSlave		
	Trigger2	BOOL	ЛОЖЬ	Условие выполнения 2
	LastTrigger2	BOOL	ЛОЖЬ	Значение *Trigger2* в предыдущем цикле выполнения задачи
	Operating2Start	BOOL	ЛОЖЬ	Обработка 2 началась.
	Operating2	BOOL	ЛОЖЬ	Обработка 2
	ConnectSet	BOOL	ЛОЖЬ	Идет выполнение команды EC_ConnectSlave.
	ConnectReset	BOOL	ЛОЖЬ	Команда EC_ConnectSlave в данный момент не выполняется.
	EC_ConnectSlave_instance	EC_ConnectSlave		
	R_TRIG_instance1	R_TRIG		
	F_TRIG_instance1	F_TRIG		
	RS_instance1	RS		
	R_TRIG_instance2	R_TRIG		
	F_TRIG_instance2	F_TRIG		
	RS_instance2	RS		
	HMI_ConnectErrorID*1	WORD	16#0000	

*1. Переменные, имя которых начинается с *HMI_*, являются переменными для отображения на сенсорной панели.

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EC_EntrySlavTbl	ARRAY[1..512] OF BOOL*1	☑	Таблица ведомых устройств, подключенных к сети
	_EC_DisconnSlavTbl	ARRAY[1..512] OF BOOL*1	☑	Таблица отсоединенных ведомых устройств
	ExclusiveFlg	BOOL		Флаг исключения одновременного выполнения команд

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

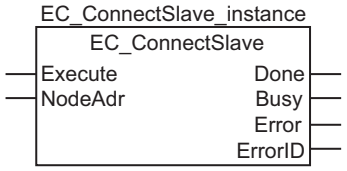
```
// Определение перехода Trigger1 в состояние ИСТИНА
IF ( (Trigger1=TRUE) AND (LastTrigger1=FALSE) AND (_EC_EntrySlavTbl[1]=TRUE) ) THEN
  Operating1Start:=TRUE;
  Operating1 :=TRUE;
END_IF;
LastTrigger1:=Trigger1;

// Инициализация команды EC_DisconnectSlave
IF (Operating1Start=TRUE) THEN
  EC_DisconnectSlave_instance(Execute:=FALSE);
  Operating1Start:=FALSE;
END_IF;
```

```
// Выполнение команды EC_DisconnectSlave
IF (Operating1=TRUE) THEN
  EC_DisconnectSlave_instance(
    Execute:=NOT(ExclusiveFlg),
    NodeAdr:=UINT#1);

  // Исключение одновременного выполнения команд
  R_TRIG_instance1(EC_DisconnectSlave_instance.Busy, DisconnectSet);
  F_TRIG_instance1(EC_DisconnectSlave_instance.Busy, DisconnectReset);
  RS_instance1(DisconnectSet, DisconnectReset, ExclusiveFlg);

  IF (EC_DisconnectSlave_instance.Done=TRUE) THEN
    // Обработка после нормального завершения
    Operating1:=FALSE;
  END_IF;

  IF (EC_DisconnectSlave_instance.Error=TRUE) THEN
    // Обработка после завершения с ошибкой
    Operating1:=FALSE;
  END_IF;
END_IF;

// Определение перехода Trigger2 в состояние ИСТИНА
IF ( (Trigger2=TRUE) AND (LastTrigger2=FALSE) AND (_EC_DisconnSlavTbl[1]=TRUE) AND
(_EC_EntrySlavTbl[1]=TRUE)) THEN
  Operating2Start:=TRUE;
  Operating2 :=TRUE;
END_IF;
LastTrigger2:=Trigger2;

// Инициализация команды EC_ConnectSlave
IF (Operating2Start=TRUE) THEN
  EC_ConnectSlave_instance(Execute:=FALSE);
  Operating2Start:=FALSE;
END_IF;

// Выполнение команды EC_ConnectSlave
IF (Operating2=TRUE) THEN
  EC_ConnectSlave_instance(
    Execute:=NOT(ExclusiveFlg),
  NodeAdr:=UINT#1);

  // Исключение одновременного выполнения команд
  R_TRIG_instance2(EC_ConnectSlave_instance.Busy, ConnectSet);
  F_TRIG_instance2(EC_ConnectSlave_instance.Busy, ConnectReset);
  RS_instance2(ConnectSet, ConnectReset, ExclusiveFlg);
  IF (EC_ConnectSlave_instance.Done=TRUE) THEN
    // Обработка после нормального завершения
    HMI_ConnectErrorID:=EC_ConnectSlave_instance.ErrorID;
    Operating2:=FALSE;
  END_IF;
```

```
  IF (EC_ConnectSlave_instance.Error=TRUE) THEN
    // Обработка после завершения с ошибкой
    HMI_ConnectErrorID:=EC_ConnectSlave_instance.ErrorID;
    Operating2:=FALSE;
  END_IF;
END_IF;
```

EC_ConnectSlave

Команда EC_ConnectSlave подсоединяет указанное ведомое устройство к сети EtherCAT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EC_ConnectSlave	Подсоединение ведомого устройства EtherCAT	FB		EC_ConnectSlave_instance(Execute, NodeAdr, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
NodeAdr	Адрес узла ведомого устройства	Вход	Адрес узла ведомого устройства для подключения	0*1...512*2	---	---

*1. Значение *0* означает, что указаны все ведомые устройства, которые зарегистрированы в настройках сети.
*2. Допустимый диапазон для модулей ЦПУ NX102, NX1P2 и модуля ЦПУ серии NJ: *0...192*.

Информация о версии

Для модуля ЦПУ серии NJ допустимый диапазон адресов узлов ведомых устройств зависит от версии следующим образом:
- Версия 1.10 или более поздняя: *0...192*
- Версия 1.09 или более ранняя: *1...192*

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NodeAdr							OK													

Функция

Команда EC_ConnectSlave подсоединяет к сети EtherCAT ведомое устройство, указанное параметром *NodeAdr* (адрес узла ведомого устройства).
В данном случае под подсоединением к сети подразумевается, что ведомое устройство присутствует в сети и переведено в состояние, в котором оно работает.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_EntrySlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, подключенных к сети	BOOL[]	Эта переменная показывает, являются ли ведомые устройства частью сети (т. е. присутствуют ли они в ней). ИСТИНА: является частью сети. ЛОЖЬ: не является частью сети.
_EC_DisconnSlavTbl[i] "i" — это адрес узла.	Таблица отсоединенных ведомых устройств	BOOL[]	Эта переменная показывает ведомые устройства, для которых в настоящее время действуют команды отсоединения от сети. ИСТИНА: действует команда отсоединения. ЛОЖЬ: не действует команда отсоединения.

Дополнительная информация

Дополнительные сведения об интерфейсе связи EtherCAT см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя (Cat. No. W505)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать только для портов EtherCAT Серия NJ/NX.
- Эту команду невозможно выполнить во время выполнения следующих команд: EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, ResetECError, RestartNXUnit и NX_ChangeWriteMode.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки.
  a) Ведомое устройство с указанным адресом узла *NodeAdr* не является частью сети EtherCAT. То есть элемент _EC_EntrySlavTbl[i] (Таблица ведомых устройств, подключенных к сети) для этого устройства содержит значение ЛОЖЬ.
  b) Уже выполняется команда EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, ResetECError, RestartNXUnit или NX_ChangeWriteMode.
  c) Было выполнено более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.

Пример программы

См. *Пример программы* на стр. 2-1084 для команды EC_DisconnectSlave.

EC_ChangeEnableSetting

Команда EC_ChangeEnableSetting активирует или деактивирует ведомое устройство EtherCAT.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
EC_ChangeEnableSetting	Активация/ деактивация ведомого устройства EtherCAT	FB	EC_ChangeEnableSetting_instance EC_ChangeEnableSetting Execute, NodeAdr, IsEnable Done, Busy, Error, ErrorID	EC_ChangeEnableSetting_instance(Execute, NodeAdr, IsEnable, Done, Busy, Error, ErrorID);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.04 или более поздней и Sysmac Studio версии 1.05 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
NodeAdr	Адрес узла ведомого устройства	Вход	Адрес узла активируемого или деактивируемого ведомого устройства EtherCAT	1...512*1	---	1
IsEnable	Выбор активации/ деактивации		Выбор активации или деактивации указанного ведомого устройства EtherCAT ИСТИНА: активировать ЛОЖЬ: деактивировать	Зависит от типа данных.		ИСТИНА

*1. Допустимый диапазон для модулей ЦПУ NX102, NX1P2 и модуля ЦПУ серии NJ: *1...192*.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NodeAdr							ОК													
IsEnable	ОК																			

Функция

Команда EC_ChangeEnableSetting активирует или деактивирует ведомое устройство EtherCAT, указанное параметром *NodeAdr* (адрес узла ведомого устройства).

Ведомое устройство активируется, если в параметр *IsEnable* (выбор активации/деактивации) передано значение ИСТИНА, или деактивируется — если передано значение ЛОЖЬ.
В случае успешного завершения данной команды ее выход *Done* переходит в состояние ИСТИНА.
Активация или деактивация ведомого устройства происходит в случае нормального завершения команды.

Команда может не завершиться успешно в зависимости от состояния указанного ведомого устройства EtherCAT: активировано ли указанное ведомое устройство EtherCAT или деактивировано, подсоединено ли оно к сети или отсоединено, присутствует ли оно в сети EtherCAT или отсутствует.

В следующей таблице показано, как изменяется состояние ведомого устройства EtherCAT после выполнения этой команды.

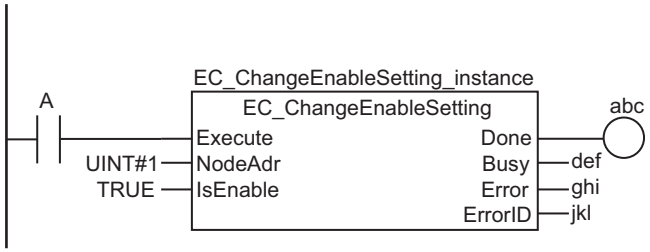
Состояние до выполнения команды			Значение *IsEnable*	Состояние после выполнения команды	
Активировано/деактивировано	**Подсоединено/отсоединено**	**Присутствует**[*1]		**Нормальное завершение/завершение с ошибкой**	**Активировано/деактивировано**
Активировано	Подсоединено	Да	ИСТИНА (подсоединено)	Нормальное завершение	Активировано
	Отсоединено	Да		Завершение с ошибкой[*2]	Активировано
		Нет			
Деактивировано	---[*3]	Да		Нормальное завершение	Активировано
		Нет		Завершение с ошибкой[*4]	Деактивировано[*4]
Активировано	Подсоединено	Да	ЛОЖЬ (деактивировано)	Нормальное завершение	Деактивировано
	Отсоединено	Да		Завершение с ошибкой[*2]	Активировано
		Нет			
Деактивировано	---[*3]	Да		Нормальное завершение	Деактивировано
		Нет		Завершение с ошибкой[*4]	Деактивировано[*4]

*1. Указывает, подключено ли указанное ведомое устройство EtherCAT к сети EtherCAT физически.
Да: устройство физически подключено. Нет: устройство не подключено физически.
*2. Для проектной версии модуля 1.40 или более поздней версии возвращается код ошибки 180A.
Для проектной версии модуля ранее версии 1.40 возвращается код ошибки 1800.
*3. Деактивированные ведомые устройства EtherCAT не считаются ни подсоединенными, ни отсоединенными.
*4. Команда завершается с ошибкой, сохраняется состояние активации/деактивации до выполнения команды, возвращается код ошибки 1801.

● Пример применения

Представленный ниже пример демонстрирует активацию ведомого устройства EtherCAT с адресом узла 1.
В параметр *NodeAdr* передается значение UINT#1, а в параметр *IsEnable* — значение ИСТИНА.

LD



ST

```
EC_ChangeEnableSetting_instance(A, UINT#1,
                                TRUE, abc, def, ghi, jkl);
```

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_EntrySlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, подключенных к сети	BOOL[]	Эта переменная показывает, являются ли ведомые устройства частью сети (т. е. присутствуют ли они в ней). ИСТИНА: является частью сети. ЛОЖЬ: не является частью сети.
_EC_DisconnSlavTbl[i] "i" — это адрес узла.	Таблица отсоединенных ведомых устройств	BOOL[]	Эта переменная показывает ведомые устройства, для которых в настоящее время действуют команды отсоединения от сети. ИСТИНА: действует команда отсоединения. ЛОЖЬ: не действует команда отсоединения.
_EC_DisableSlavTbl[i] "i" — это адрес узла.	Таблица деактивированных ведомых устройств	BOOL[]	Эта переменная показывает, не деактивировано ли то или иное ведомое устройство в сети. ИСТИНА: деактивировано. ЛОЖЬ: не деактивировано или не является частью сети.

Дополнительная информация

- Дополнительные сведения об интерфейсе связи EtherCAT см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherCAT. Руководство пользователя (Cat. No. W505)*.
- Для подсоединения ведомого устройства EtherCAT к сети EtherCAT используйте команду *EC_ConnectSlave* на стр. 2-1091.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать только для портов EtherCAT Серия NJ/NX.
- Эту команду невозможно выполнить во время выполнения следующих команд: EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, ResetECError, RestartNXUnit и NX_ChangeWriteMode.
- Результаты выполнения этой команды не сохраняются в энергонезависимую память модуля ЦПУ. Следовательно, если после выполнения этой команды будет выключено и вновь

включено питание контроллера или будет загружена программа пользователя, ведомое устройство EtherCAT вернется в состояние активации/деактивации, заданное для него в Sysmac Studio.

- Допускается одновременное выполнение не более 32 экземпляров следующих команд: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА, а переменной *ErrorID* будет присвоен код ошибки.
  a) Ведомое устройство с указанным адресом узла *NodeAdr* не является частью сети EtherCAT. То есть элемент _EC_EntrySlavTbl[i] (Таблица ведомых устройств, подключенных к сети) для этого устройства содержит значение ЛОЖЬ.
  b) Значение *NodeAdr* находится за пределами допустимого диапазона.
  c) Уже выполняется команда EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, ResetECError, RestartNXUnit или NX_ChangeWriteMode.
  d) Было выполнено более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
  e) Команда выполнена для ведомого устройства в сети с кольцевой топологией.
  Ошибка в этом случае возникает независимо от того, в каком состоянии находится сеть во время выполнения команды: в состоянии резервирования кабельных соединений или в состоянии разрыва кольца.

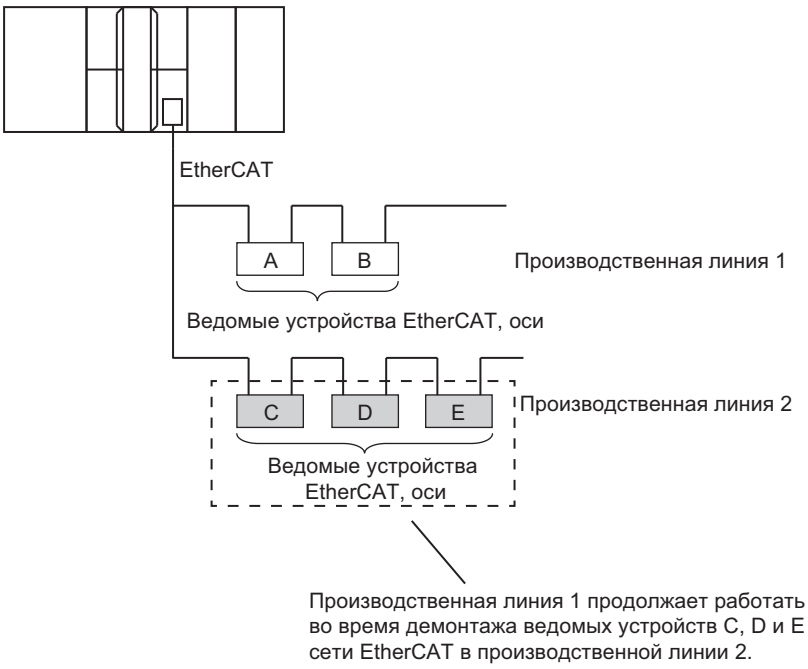
Пример программы

В этом разделе будет рассмотрено два следующих примера, поясняющих работу команды:

- Пример отсоединения ведомых устройств EtherCAT от сети EtherCAT
- Пример подсоединения ведомых устройств EtherCAT к сети EtherCAT

Пример отсоединения ведомых устройств EtherCAT от сети EtherCAT

Производственная линия 1 системы (см. рис. ниже) продолжает работать, в то время как ведомые устройства EtherCAT C, D и E производственной линии 2 демонтируются.
Для ведомых устройств EtherCAT C, D и E уже настроены оси управления движением. Поэтому ведомые устройства EtherCAT деактивируются, а настроенные для них оси переводятся в состояние «неиспользуемая ось».



● Порядок действий

Рассматриваемая в данном примере программа реализует следующий порядок действий:

1 Оператор нажимает кнопку на операторской панели, тем самым активируя условие выполнения.

2 Контроллер деактивирует ведомые устройства EtherCAT C, D и E. Кроме того, настроенные для этих ведомых устройств оси переводятся в состояние «неиспользуемая ось».

3 После того как все три ведомых устройства деактивированы, а их оси сделаны неиспользуемыми, контроллер включает лампу «Демонтаж разрешен».

4 Увидев, что лампа «Демонтаж разрешен» горит, оператор удаляет из системы три ведомых устройства EtherCAT.

● Команда для перевода осей в состояние «неиспользуемая ось»

Для перевода осей в состояние «неиспользуемая ось» служит команда MC_ChangeAxisUse. Подробное описание команды MC_ChangeAxisUse см. в документе *Серия NJ/NX — Команды программирования для управления движением. Справочное руководство (Cat. No. W508)*.

● Исключение одновременного выполнения команд

В любой момент времени может выполняться только один экземпляр команды EC_ChangeEnableSetting. Несколько экземпляров команды EC_ChangeEnableSetting одновременно выполняться не могут.

Кроме того, команда EC_ChangeEnableSetting выполняется дольше одного периода выполнения задачи.

Прежде чем выполнять очередную команду EC_ChangeEnableSetting, необходимо удостовериться в том, что завершилось выполнение предыдущей команды EC_ChangeEnableSetting.

Для этой цели используется переменная *ExclusiveFlg* (Флаг исключения одновременного выполнения команд).
Если флаг *ExclusiveFlg* = *И*СТИНА, значит команда EC_ChangeEnableSetting в данный момент выполняется.
Пока значение флага *ExclusiveFlg* = *И*СТИНА, следующую команду EC_ChangeEnableSetting выполнять не следует.

Команду EC_ChangeEnableSetting невозможно выполнить, если в это время в другой задаче выполняется другой экземпляр команды EC_ChangeEnableSetting.
Поэтому в рассматриваемом примере флаг *ExclusiveFlg* определен как глобальная переменная. Благодаря этому в программе можно реализовать эксклюзивное управление, чтобы исключить одновременное выполнение команды EC_ChangeEnableSetting в других задачах.
Эта же глобальная переменная *ExclusiveFlg* должна использоваться и в других задачах для той же цели (т. е. для исключения одновременного выполнения команд).
Команду EC_ChangeEnableSetting нельзя выполнять одновременно с командой EC_DisconnectSlave или EC_ConnectSlave.
Эта же глобальная переменная *ExclusiveFlg* используется в разделе *Пример программы* на стр. 2-1084 для команды EC_DisconnectSlave для пояснения работы эксклюзивного управления командами.

● Переменные осей и адреса узлов для ведомых устройств EtherCAT

В следующей таблице указаны переменные осей, которые назначены осям для ведомых устройств EtherCAT C, D и E, а также адреса узлов ведомых устройств.

Ведомые устройства EtherCAT	Переменная оси	Адрес узла
C	MC_Axis000	1
D	MC_Axis001	2
E	MC_Axis002	3

● Глобальные переменные

Переменная	Тип данных	Начальное значение	Параметр «AT»	Константа	Комментарий
MC_Axis000	_sAXIS_REF		MC:// _MC_AX[0]	☑	Переменная оси для ведомого устройства EtherCAT C
MC_Axis001	_sAXIS_REF		MC:// _MC_AX[1]	☑	Переменная оси для ведомого устройства EtherCAT D
MC_Axis002	_sAXIS_REF		MC:// _MC_AX[2]	☑	Переменная оси для ведомого устройства EtherCAT E
ExclusiveFlg	BOOL	ЛОЖЬ		☐	Флаг исключения одновременного выполнения команд

● Программа на языке LD

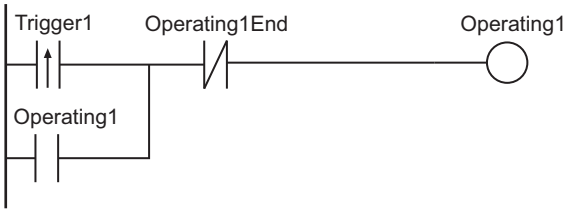
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Operating1End	BOOL	ЛОЖЬ	Обработка завершена
	Trigger1	BOOL	ЛОЖЬ	Условие выполнения
	Operating1	BOOL	ЛОЖЬ	Обработка

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	AxisUnuseDone_DevC	BOOL	ЛОЖЬ	Перевод оси в состояние «неиспользуемая ось» завершен для ведомого устройства EtherCAT C
	SlaveDisableDone_DevC	BOOL	ЛОЖЬ	Деактивация ведомого устройства EtherCAT C завершена
	DoneHold_DevC	BOOL	ЛОЖЬ	Флаг завершения обработки для ведомого устройства EtherCAT C
	AxisUnuseDone_DevD	BOOL	ЛОЖЬ	Перевод оси в состояние «неиспользуемая ось» завершен для ведомого устройства EtherCAT D
	SlaveDisableDone_DevD	BOOL	ЛОЖЬ	Деактивация ведомого устройства EtherCAT D завершена
	DoneHold_DevD	BOOL	ЛОЖЬ	Флаг завершения обработки для ведомого устройства EtherCAT D
	AxisUnuseDone_DevE	BOOL	ЛОЖЬ	Перевод оси в состояние «неиспользуемая ось» завершен для ведомого устройства EtherCAT E
	SlaveDisableDone_DevE	BOOL	ЛОЖЬ	Деактивация ведомого устройства EtherCAT E завершена
	DoneHold_DevE	BOOL	ЛОЖЬ	Флаг завершения обработки для ведомого устройства EtherCAT E
	Light1On	BOOL	ЛОЖЬ	Включение лампы «Демонтаж разрешен»
	MC_ChangeAxisUse_DevC	MC_ChangeAxisUse		
	EC_ChangeEnableSetting_DevC	EC_ChangeEnableSetting		
	MC_ChangeAxisUse_DevD	MC_ChangeAxisUse		
	EC_ChangeEnableSetting_DevD	EC_ChangeEnableSetting		
	MC_ChangeAxisUse_DevE	MC_ChangeAxisUse		
	EC_ChangeEnableSetting_DevE	EC_ChangeEnableSetting		

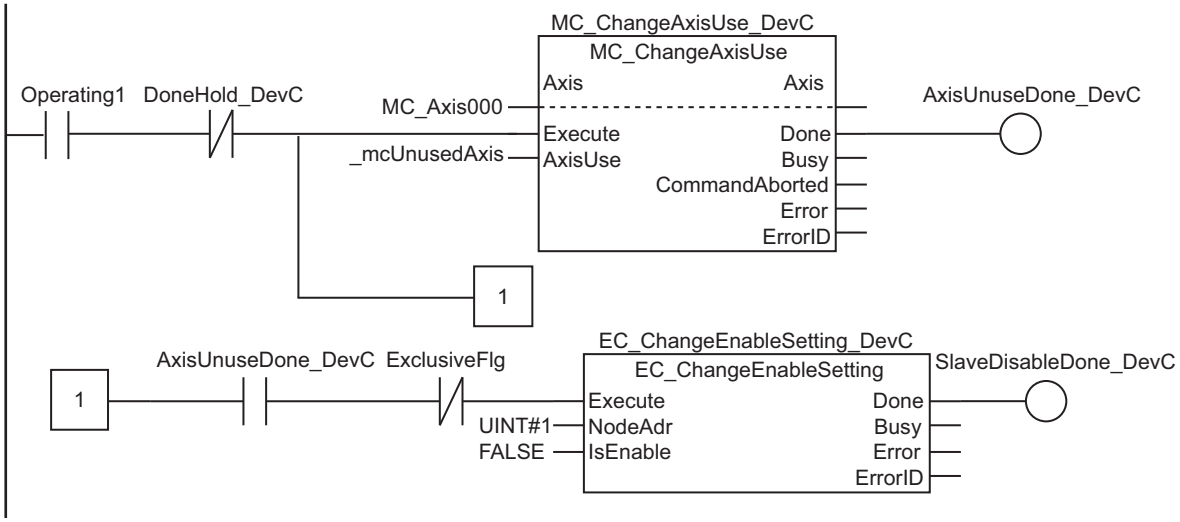
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	MC_Axis000	_sAXIS_REF	☑	Переменная оси для ведомого устройства EtherCAT C
	MC_Axis001	_sAXIS_REF	☑	Переменная оси для ведомого устройства EtherCAT D

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	MC_Axis002	_sAXIS_REF	☑	Переменная оси для ведомого устройства EtherCAT E
	ExclusiveFlg	BOOL	☐	Флаг исключения одновременного выполнения команд

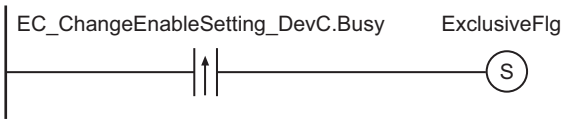
Прием сигнала условия выполнения.

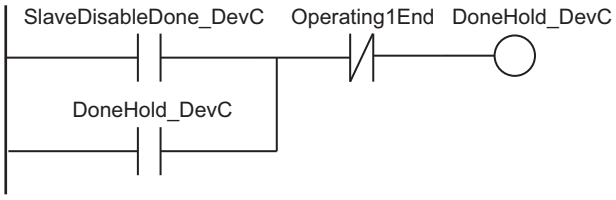


Перевод оси в состояние «неиспользуемая ось» и деактивация ведомого устройства С сети EtherCAT.

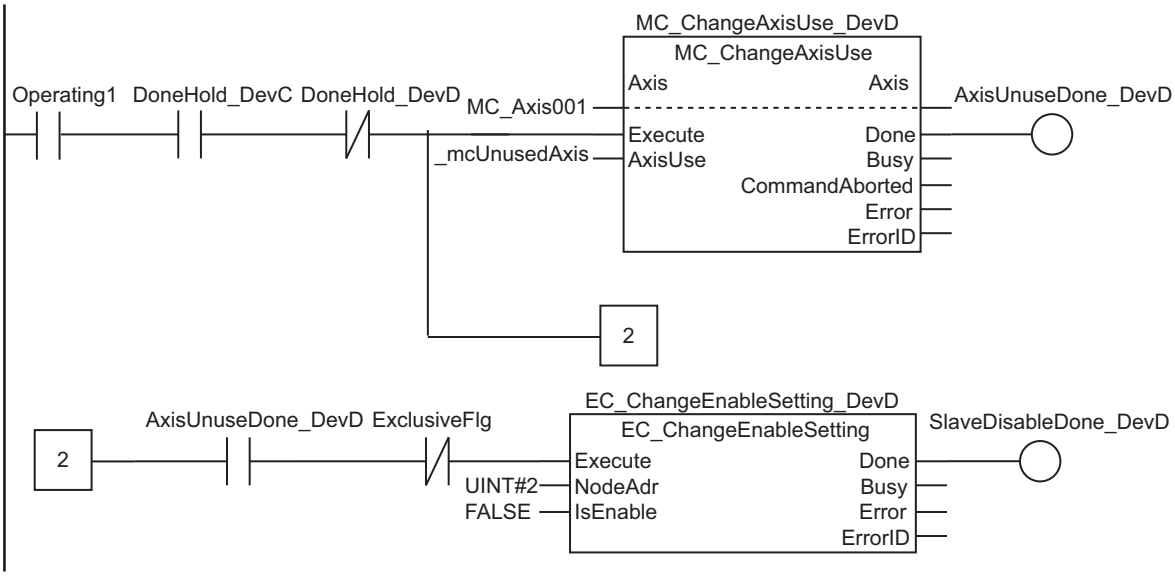


Исключение одновременного выполнения команд

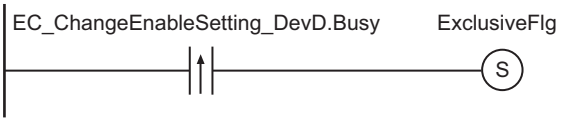


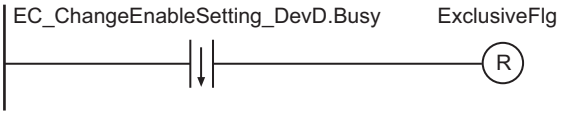


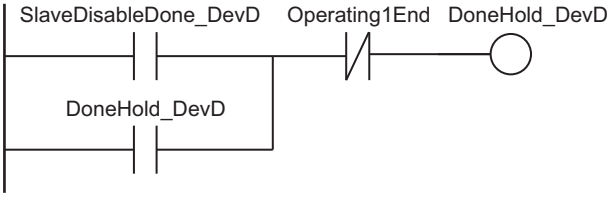
Перевод оси в состояние «неиспользуемая ось» и деактивация ведомого устройства D сети EtherCAT.



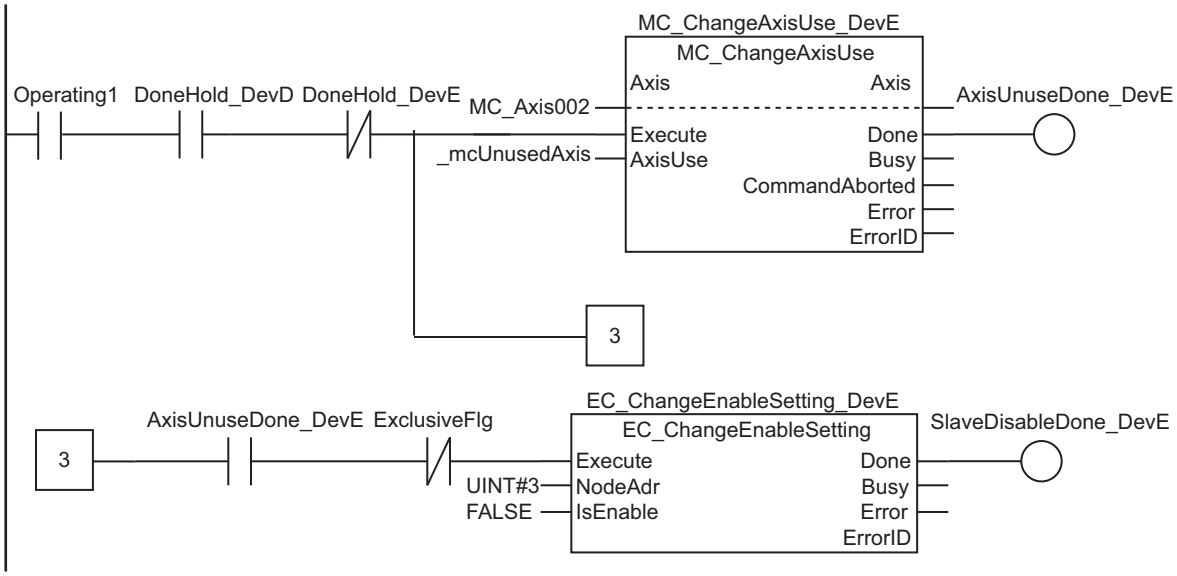
Исключение одновременного выполнения команд



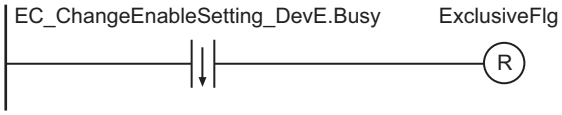


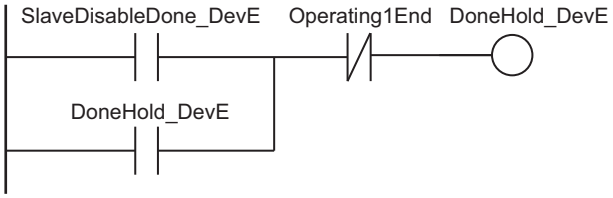


Перевод оси в состояние «неиспользуемая ось» и деактивация ведомого устройства E сети EtherCAT.



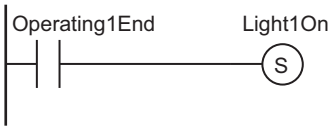
Исключение одновременного выполнения команд





Перевод оси в состояние «неиспользуемая ось» и деактивация ведомого устройства E сети EtherCAT.

Включение лампы «Демонтаж разрешен»



● Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Operating1End	BOOL	ЛОЖЬ	Обработка завершена
	Trigger1	BOOL	ЛОЖЬ	Условие выполнения
	Operating1	BOOL	ЛОЖЬ	Обработка
	Operating1Set	BOOL	ЛОЖЬ	Обработка началась
	Light1On	BOOL	ЛОЖЬ	Включение лампы «Демонтаж разрешен»
	DoneHold_DevC	BOOL	ЛОЖЬ	Флаг завершения обработки для ведомого устройства EtherCAT C
	DoneHold_DevD	BOOL	ЛОЖЬ	Флаг завершения обработки для ведомого устройства EtherCAT D
	DoneHold_DevE	BOOL	ЛОЖЬ	Флаг завершения обработки для ведомого устройства EtherCAT E
	ExclusiveFlgSet	BOOL	ЛОЖЬ	Установка флага исключения одновременного выполнения команд
	ExclusiveFlgReset	BOOL	ЛОЖЬ	Сброс флага исключения одновременного выполнения команд
	R_TRIG_instance1	R_TRIG		
	RS_instance1	RS		
	SR_instance1	SR		
	MC_ChangeAxisUse_DevC	MC_ChangeAxisUse		
	EC_ChangeEnableSetting_DevC	EC_ChangeEnableSetting		
	R_TRIG_DevC	R_TRIG		
	F_TRIG_DevC	F_TRIG		
	RS_ExFlg_DevC	RS		
	RS_DevC	RS		
	MC_ChangeAxisUse_DevD	MC_ChangeAxisUse		
	EC_ChangeEnableSetting_DevD	EC_ChangeEnableSetting		
	R_TRIG_DevD	R_TRIG		
	F_TRIG_DevD	F_TRIG		
	RS_ExFlg_DevD	RS		
	RS_DevD	RS		
	MC_ChangeAxisUse_DevE	MC_ChangeAxisUse		
	EC_ChangeEnableSetting_DevE	EC_ChangeEnableSetting		
	R_TRIG_DevE	R_TRIG		
	F_TRIG_DevE	F_TRIG		
	RS_ExFlg_DevE	RS		
	RS_DevE	RS		

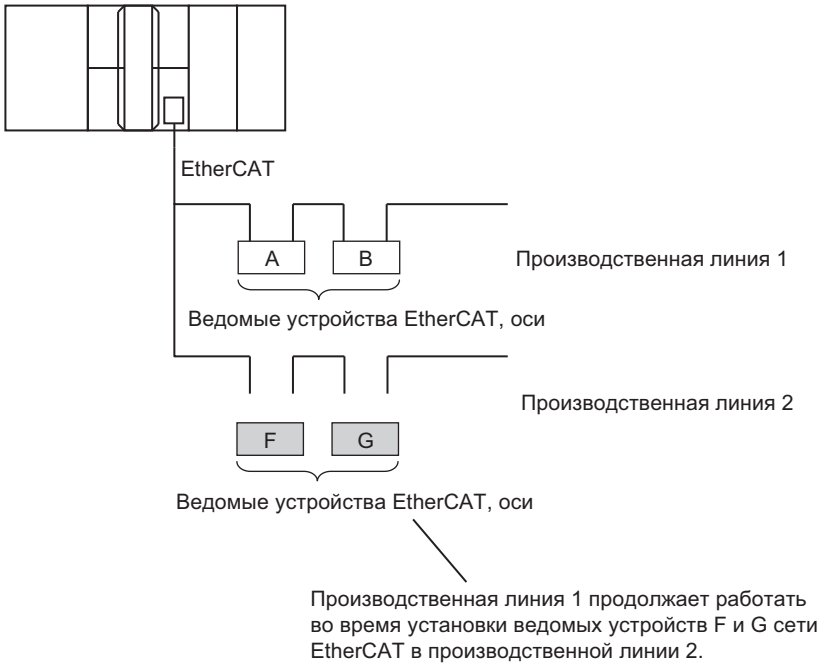
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	MC_Axis000	_sAXIS_REF	☑	Переменная оси для ведомого устройства EtherCAT C
	MC_Axis001	_sAXIS_REF	☑	Переменная оси для ведомого устройства EtherCAT D
	MC_Axis002	_sAXIS_REF	☑	Переменная оси для ведомого устройства EtherCAT E
	ExclusiveFlg	BOOL	☐	Флаг исключения одновременного выполнения команд

```
// Прием условия выполнения.
R_TRIG_instance1(Trigger1, Operating1Set);
RS_instance1(
  Set :=Operating1Set,
  Reset1:=Operating1End,
  Q1 =>Operating1);

// Перевод оси в состояние «неиспользуемая ось» для ведомого устройства EtherCAT C.
MC_ChangeAxisUse_DevC(
  Axis :=MC_Axis000,
  Execute:=(Operating1 & NOT(DoneHold_DevC)),
  AxisUse:=_mcUnusedAxis);

// Деактивация ведомого устройства EtherCAT C.
EC_ChangeEnableSetting_DevC(
  Execute :=(Operating1 & MC_ChangeAxisUse_DevC.Done & NOT(ExclusiveFlg)),
  NodeAdr :=UINT#1,
  IsEnable:=FALSE);

// Исключение одновременного выполнения команд
R_TRIG_DevC(EC_ChangeEnableSetting_DevC.Busy, ExclusiveFlgSet);
F_TRIG_DevC(EC_ChangeEnableSetting_DevC.Busy, ExclusiveFlgReset);
RS_ExFlg_DevC(
  Set :=ExclusiveFlgSet,
  Reset1:=ExclusiveFlgReset,
  Q1 =>ExclusiveFlg);
RS_DevC(
  Set :=EC_ChangeEnableSetting_DevC.Done,
  Reset1:=Operating1End,
  Q1 =>DoneHold_DevC);

// Перевод оси в состояние «неиспользуемая ось» для ведомого устройства EtherCAT D.
MC_ChangeAxisUse_DevD(
  Axis :=MC_Axis001,
  Execute:=(Operating1 & DoneHold_DevC & NOT(DoneHold_DevD)),
  AxisUse:=_mcUnusedAxis);

// Деактивация ведомого устройства EtherCAT D.
```

```
EC_ChangeEnableSetting_DevD(
  Execute :=(Operating1 & DoneHold_DevC & MC_ChangeAxisUse_DevD.Done & NOT(Exclusiv
eFlg)),
  NodeAdr :=UINT#2,
  IsEnable:=FALSE);

// Исключение одновременного выполнения команд
R_TRIG_DevD(EC_ChangeEnableSetting_DevD.Busy, ExclusiveFlgSet);
F_TRIG_DevD(EC_ChangeEnableSetting_DevD.Busy, ExclusiveFlgReset);
RS_ExFlg_DevD(
  Set :=ExclusiveFlgSet,
  Reset1:=ExclusiveFlgReset,
  Q1 =>ExclusiveFlg);
RS_DevD(
  Set :=EC_ChangeEnableSetting_DevD.Done,
  Reset1:=Operating1End,
  Q1 =>DoneHold_DevD);

// Перевод оси в состояние «неиспользуемая ось» для ведомого устройства EtherCAT E.
MC_ChangeAxisUse_DevE(
  Axis :=MC_Axis002,
  Execute:=(Operating1 & DoneHold_DevD & NOT(DoneHold_DevE)),
  AxisUse:=_mcUnusedAxis);

// Деактивация ведомого устройства EtherCAT E.
EC_ChangeEnableSetting_DevE(
  Execute :=(Operating1 & DoneHold_DevD & MC_ChangeAxisUse_DevE.Done & NOT(Exclusiv
eFlg)),
  NodeAdr :=UINT#3,
  IsEnable:=FALSE);

// Исключение одновременного выполнения команд
R_TRIG_DevE(EC_ChangeEnableSetting_DevE.Busy, ExclusiveFlgSet);
F_TRIG_DevE(EC_ChangeEnableSetting_DevE.Busy, ExclusiveFlgReset);
RS_ExFlg_DevE(
  Set :=ExclusiveFlgSet,
  Reset1:=ExclusiveFlgReset,
  Q1 =>ExclusiveFlg);
RS_DevE(
  Set :=EC_ChangeEnableSetting_DevE.Done,
  Reset1:=Operating1End,
  Q1 =>DoneHold_DevE);

// Подтверждение перевода оси в состояние «неиспользуемая ось» и деактивации ведомо
го устройства EtherCAT E.
Operating1End:=(Operating1 & DoneHold_DevE);

// Включение лампы «Демонтаж разрешен»
SR_instance1(
  Set1:=Operating1End,
  Q1 =>Light1On);
```

Пример подсоединения ведомых устройств EtherCAT к сети EtherCAT

Производственная линия 1 из предыдущего примера продолжает работать, в то время как в производственной линии 2 устанавливаются ведомые устройства EtherCAT F и G.
Для ведомых устройств EtherCAT F и G настроены оси управления движением.
Поэтому ведомые устройства EtherCAT активируются, а настроенные для них оси переводятся в состояние «используемая ось».



● Порядок действий

Рассматриваемая в данном примере программа реализует следующий порядок действий:

1 Оператор устанавливает ведомые устройства EtherCAT F и G, соблюдая указанный ниже порядок действий.

2 Оператор нажимает кнопку на операторской панели, тем самым активируя условие выполнения.

3 Контроллер активирует ведомые устройства EtherCAT F и G. Кроме того, настроенные для этих ведомых устройств оси переводятся в состояние «используемая ось».

4 После того как оба ведомых устройства EtherCAT активированы, а их оси сделаны используемыми, контроллер включает лампу «Установка завершена».

● Команда для перевода осей в состояние «используемая ось»

Для перевода осей в состояние «используемая ось» служит команда MC_ChangeAxisUse.
Подробное описание команды MC_ChangeAxisUse см. в документе *Серия NJ/NX — Команды программирования для управления движением. Справочное руководство (Cat. No. W508)*.

● Исключение одновременного выполнения команд

В любой момент времени может выполняться только один экземпляр команды EC_ChangeEnableSetting. Несколько экземпляров команды EC_ChangeEnableSetting одновременно выполняться не могут.
Кроме того, команда EC_ChangeEnableSetting выполняется дольше одного периода выполнения задачи.
Прежде чем выполнять очередную команду EC_ChangeEnableSetting, необходимо удостовериться в том, что завершилось выполнение предыдущей команды EC_ChangeEnableSetting.
Для этой цели используется переменная *ExclusiveFlg* (Флаг исключения одновременного выполнения команд).
Если флаг *ExclusiveFlg* = ИСТИНА, значит команда EC_ChangeEnableSetting в данный момент выполняется.
Пока значение флага *ExclusiveFlg* = ИСТИНА, следующую команду EC_ChangeEnableSetting выполнять не следует.

Команду EC_ChangeEnableSetting невозможно выполнить, если в это время в другой задаче выполняется другой экземпляр команды EC_ChangeEnableSetting.
В рассматриваемом примере флаг *ExclusiveFlg* определен как глобальная переменная.
Благодаря этому в программе можно реализовать эксклюзивное управление, чтобы исключить одновременное выполнение команды EC_ChangeEnableSetting в других задачах.
В данном случае, однако, эта же глобальная переменная (*ExclusiveFlg*) должна использоваться и в других задачах для той же цели (т. е. для исключения одновременного выполнения команд).
Команду EC_ChangeEnableSetting нельзя выполнять одновременно с командой EC_DisconnectSlave или EC_ConnectSlave.
Эта же глобальная переменная *ExclusiveFlg* используется в разделе *Пример программы* на стр. 2-1084 для команды EC_DisconnectSlave для пояснения работы эксклюзивного управления командами.

● Переменные осей и адреса узлов для ведомых устройств EtherCAT

В следующей таблице указаны переменные осей, которые назначены осям для ведомых устройств EtherCAT F и G, а также адреса узлов ведомых устройств.

Ведомые устройства EtherCAT	Переменная оси	Адрес узла
F	MC_Axis003	4
G	MC_Axis004	5

● Глобальные переменные

Переменная	Тип данных	Начальное значение	Параметр «AT»	Константа	Комментарий
MC_Axis003	_sAXIS_REF		MC:// _MC_AX[3]	☑	Переменная оси для ведомого устройства EtherCAT F
MC_Axis004	_sAXIS_REF		MC:// _MC_AX[4]	☑	Переменная оси для ведомого устройства EtherCAT G
ExclusiveFlg	BOOL	ЛОЖЬ		☐	Флаг исключения одновременного выполнения команд

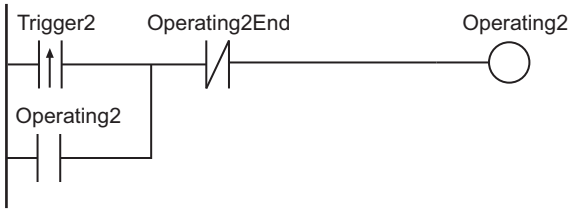
● Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Operating2End	BOOL	ЛОЖЬ	Обработка завершена
	Trigger2	BOOL	ЛОЖЬ	Условие выполнения
	Operating2	BOOL	ЛОЖЬ	Обработка
	AxisUseDone_DevF	BOOL	ЛОЖЬ	Перевод оси в состояние «используемая ось» завершен для ведомого устройства EtherCAT F
	SlaveEnableDone_DevF	BOOL	ЛОЖЬ	Активация ведомого устройства EtherCAT F завершена
	DoneHold_DevF	BOOL	ЛОЖЬ	Флаг завершения обработки для ведомого устройства EtherCAT F
	AxisUseDone_DevG	BOOL	ЛОЖЬ	Перевод оси в состояние «используемая ось» завершен для ведомого устройства EtherCAT G
	SlaveEnableDone_DevG	BOOL	ЛОЖЬ	Активация ведомого устройства EtherCAT G завершена
	DoneHold_DevG	BOOL	ЛОЖЬ	Флаг завершения обработки для ведомого устройства EtherCAT G
	Light2On	BOOL	ЛОЖЬ	Включение лампы «Установка завершена»
	MC_ChangeAxisUse_DevF	MC_ChangeAxisUse		
	EC_ChangeEnableSetting_DevF	EC_ChangeEnableSetting		
	MC_ChangeAxisUse_DevG	MC_ChangeAxisUse		
	EC_ChangeEnableSetting_DevG	EC_ChangeEnableSetting		

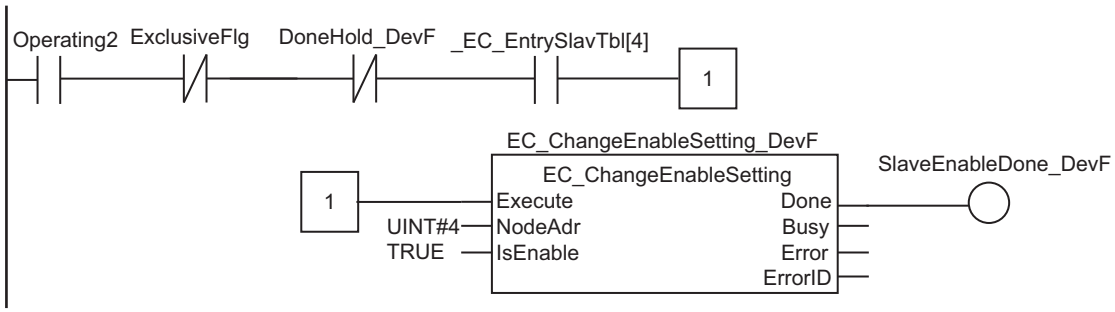
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	MC_Axis003	_sAXIS_REF	☑	Переменная оси для ведомого устройства EtherCAT F
	MC_Axis004	_sAXIS_REF	☑	Переменная оси для ведомого устройства EtherCAT G

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	ExclusiveFlg	BOOL	☐	Флаг исключения одновременного выполнения команд

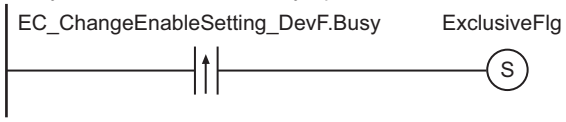
Прием сигнала условия выполнения

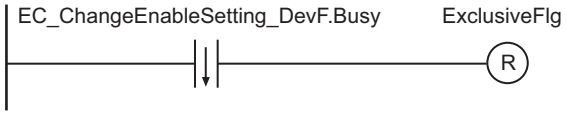


Активация ведомого устройства F сети EtherCAT

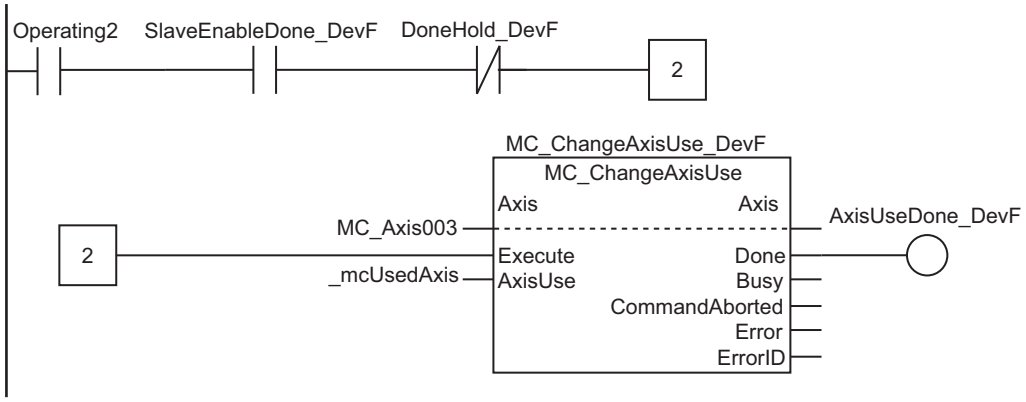


Исключение одновременного выполнения команд.
Запуск активации ведомого устройства F сети EtherCAT и подтверждение завершения.

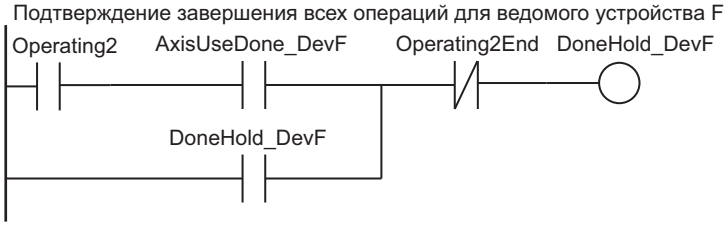




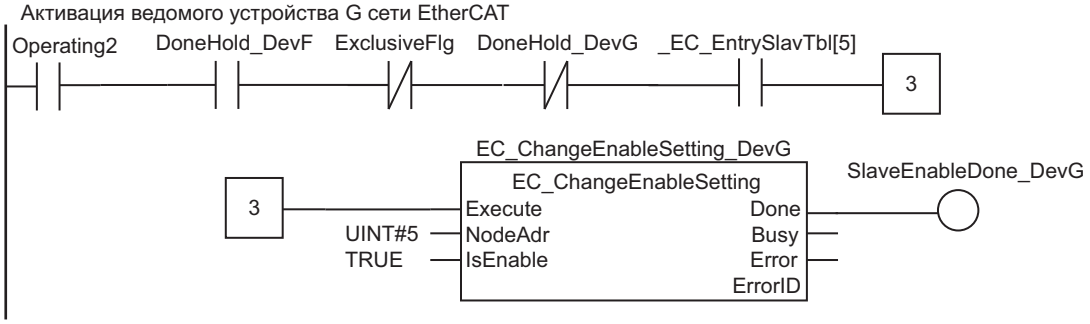
Перевод оси для ведомого устройства F сети EtherCAT в состояние «используемая ось»



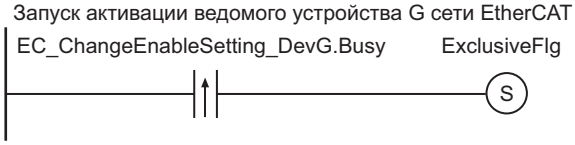
Исключение одновременного выполнения команд.
Подтверждение завершения всех операций для ведомого устройства F сети EtherCAT.

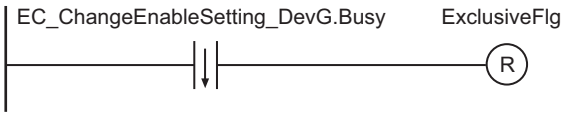


Активация ведомого устройства G сети EtherCAT

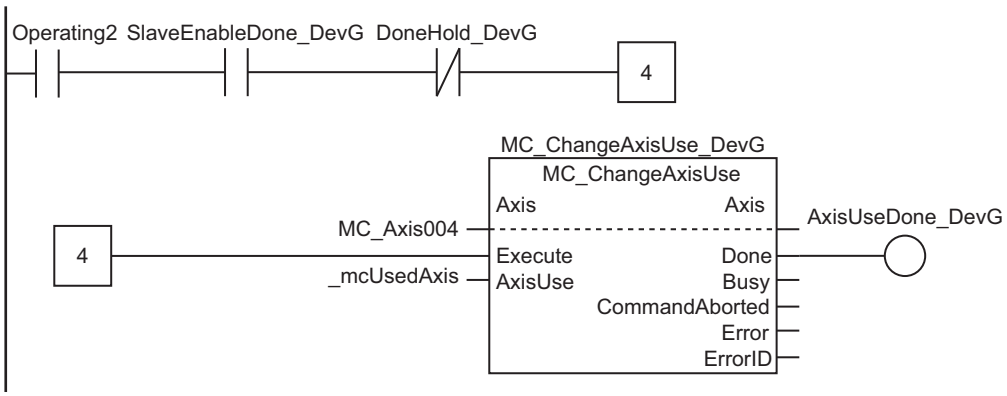


Эксклюзивное управление для исключения одновременного выполнения команд.
Запуск активации ведомого устройства G сети EtherCAT и подтверждение завершения.

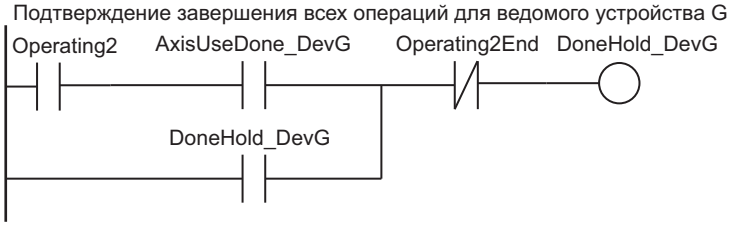




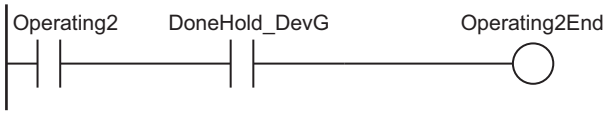
Перевод оси для ведомого устройства G сети EtherCAT в состояние «используемая ось»



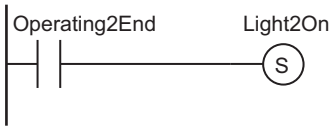
Исключение одновременного выполнения команд.
Подтверждение завершения всех операций для ведомого устройства G сети EtherCAT.



Подтверждение завершения обработки для ведомого устройства G сети EtherCAT



Включение лампы «Установка завершена»



● Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Operating2End	BOOL	ЛОЖЬ	Обработка завершена
	Trigger2	BOOL	ЛОЖЬ	Условие выполнения
	Operating2	BOOL	ЛОЖЬ	Обработка
	Operating2Set	BOOL	ЛОЖЬ	Обработка началась
	Light2On	BOOL	ЛОЖЬ	Включение лампы «Установка завершена»
	DoneHold_DevF	BOOL	ЛОЖЬ	Флаг завершения обработки для ведомого устройства EtherCAT F
	DoneHold_DevG	BOOL	ЛОЖЬ	Флаг завершения обработки для ведомого устройства EtherCAT G
	ExclusiveFlgSet	BOOL	ЛОЖЬ	Установка флага исключения одновременного выполнения команд
	ExclusiveFlgReset	BOOL	ЛОЖЬ	Сброс флага исключения одновременного выполнения команд
	R_TRIG_instance2	R_TRIG		
	RS_instance2	RS		
	SR_instance2	SR		
	MC_ChangeAxisUse_DevF	MC_ChangeAxisUse		
	EC_ChangeEnableSetting_DevF	EC_ChangeEnableSetting		
	R_TRIG_DevF	R_TRIG		
	F_TRIG_DevF	F_TRIG		
	RS_ExFlg_DevF	RS		
	RS_DevF	RS		
	MC_ChangeAxisUse_DevG	MC_ChangeAxisUse		

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	EC_ChangeEnableSetting_DevG	EC_ChangeEnableSetting		
	R_TRIG_DevG	R_TRIG		
	F_TRIG_DevG	F_TRIG		
	RS_ExFlg_DevG	RS		
	RS_DevG	RS		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	MC_Axis003	_sAXIS_REF	☑	Переменная оси для ведомого устройства EtherCAT F
	MC_Axis004	_sAXIS_REF	☑	Переменная оси для ведомого устройства EtherCAT G
	ExclusiveFlg	BOOL	☐	Флаг исключения одновременного выполнения команд

```
// Прием сигнала условия выполнения
R_TRIG_instance2(Trigger2, Operating2Set);
RS_instance2(
  Set :=Operating2Set,
  Reset1:=Operating2End,
  Q1 =>Operating2);

// Активация ведомого устройства EtherCAT F
EC_ChangeEnableSetting_DevF(
  Execute :=(Operating2 & NOT(ExclusiveFlg) & NOT(DoneHold_DevF) & _EC_EntrySlavTb
l[4]),
  NodeAdr :=UINT#4,
  IsEnable:=TRUE);

// Исключение одновременного выполнения команд. Запуск активации ведомого устройств
a EtherCAT F и подтверждение завершения
R_TRIG_DevF(EC_ChangeEnableSetting_DevF.Busy, ExclusiveFlgSet);
F_TRIG_DevF(EC_ChangeEnableSetting_DevF.Busy, ExclusiveFlgReset);
RS_ExFlg_DevF(
  Set :=ExclusiveFlgSet,
  Reset1:=ExclusiveFlgReset,
  Q1 =>ExclusiveFlg);

// Перевод оси для ведомого устройства EtherCAT F в состояние «используемая ось»
MC_ChangeAxisUse_DevF(
  Axis :=MC_Axis003,
  Execute:=(Operating2 & EC_ChangeEnableSetting_DevF.Done & NOT(DoneHold_DevF)),
  AxisUse:=_mcUsedAxis);

// Исключение одновременного выполнения команд. Подтверждение завершения всех опера
```

```
ций для ведомого устройства EtherCAT F
RS_DevF(
  Set :=(Operating2 & MC_ChangeAxisUse_DevF.Done),
  Reset1:=Operating2End,
  Q1 =>DoneHold_DevF);

// Активация ведомого устройства EtherCAT G
EC_ChangeEnableSetting_DevG(
  Execute :=(Operating2 & DoneHold_DevF & NOT(ExclusiveFlg) & NOT(DoneHold_DevG) &
    _EC_EntrySlavTbl[5]),
  NodeAdr :=UINT#5,
  IsEnable:=TRUE);

// Исключение одновременного выполнения команд. Запуск активации ведомого устройств
а EtherCAT G и подтверждение завершения
R_TRIG_DevG(EC_ChangeEnableSetting_DevG.Busy, ExclusiveFlgSet);
F_TRIG_DevG(EC_ChangeEnableSetting_DevG.Busy, ExclusiveFlgReset);
RS_ExFlg_DevG(
  Set :=ExclusiveFlgSet,
  Reset1:=ExclusiveFlgReset,
  Q1 =>ExclusiveFlg);

// Перевод оси для ведомого устройства EtherCAT G в состояние «используемая ось»
MC_ChangeAxisUse_DevG(
  Axis :=MC_Axis004,
  Execute:=(Operating2 & EC_ChangeEnableSetting_DevG.Done & NOT(DoneHold_DevG)),
  AxisUse:=_mcUsedAxis);

// Исключение одновременного выполнения команд. Подтверждение завершения всех опера
ций для ведомого устройства EtherCAT G
RS_DevG(
  Set :=(Operating2 & MC_ChangeAxisUse_DevG.Done),
  Reset1:=Operating2End,
  Q1 =>DoneHold_DevG);

// Подтверждение завершения обработки для ведомого устройства EtherCAT G
Operating2End:=Operating2 & DoneHold_DevG;

// Включение лампы «Установка завершена»
SR_instance2(
  Set1:=Operating2End,
  Q1 =>Light2On);
```

NX_WriteObj

Команда NX_WriteObj производит запись данных в объект NX в интерфейсном модуле EtherCAT или модуле NX.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_WriteObj	Запись в объект модуля NX	FB	NX_WriteObj_instance NX_WriteObj Execute / Done UnitProxy / Busy Obj / Error TimeOut / ErrorID WriteDat / ErrorIDEx	NX_WriteObj_instance(Execute, UnitProxy, Obj, TimeOut, WriteDat, Done, Busy, Error, ErrorID, ErrorIDEx);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.05 или более поздней и Sysmac Studio версии 1.06 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitProxy	Указанный модуль	Вход	Модуль, в который нужно записать данные	---	---	*1
Obj	Параметр объекта		Параметр объекта			---
TimeOut	Время ожидания		Время ожидания Если задано значение 0, время ожидания составляет 2,0 с.	0...60 000	мс	2000 (2,0 с)
WriteDat	Записываемые данные		Данные для записи в объект NX	Зависит от типа данных.	---	*1

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitProxy	Подробные сведения о структуре _sNXUNIT_ID см. в разделе *Функция* на стр. 2-1115.																			
Obj	Подробные сведения о структуре _sNXOBJ_ACCESS см. в разделе *Функция* на стр. 2-1115.																			
TimeOut							ОК													
WriteDat	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также можно указать массив.																			

Функция

Команда NX_WriteObj записывает содержимое переменной *WriteDat* в объект NX в интерфейсном модуле EtherCAT; модуле NX, подключенном к интерфейсному модулю EtherCAT; или модуле NX, подключенном к шине NX модуля ЦПУ.
Модуль, для которого нужно записать данные, указывается параметром *UnitProxy*.

Параметр *TimeOut* задает время ожидания.
Если ответ не возвращается в течение времени ожидания, считается, что произошел сбой связи. В этом случае данные не записываются.

Для переменной *UnitProxy* используется структурный тип данных _sNXUNIT_ID. Значения членов структуры приведены в таблице ниже.

Имя		Значение	Содержание	Тип данных
UnitProxy		Указанный модуль	Указанный модуль	_sNXUNIT_ID
	NodeAdr	Адрес узла	Адрес узла интерфейсного модуля	UINT
	IPAdr	IP-адрес	IP-адрес интерфейсного модуля	BYTE[5]
	UnitNo	Номер модуля	Номер модуля указанного модуля	UDINT
	Path	Путь	Информация о пути к указанному модулю	BYTE[64]
	PathLength	Допустимая длина пути *Path*	Допустимая длина пути *Path*	USINT

В *UnitProxy* следует передать переменную устройства, которая назначена указанному модулю.

Для переменной *Obj* используется структурный тип данных _sNXOBJ_ACCESS. Значения членов структуры приведены в таблице ниже.

Имя		Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Obj		Параметр объекта	Параметр объекта	_sNXOBJ_ACCESS	---	---	---
	Index	Индекс	Индекс	UINT	Зависит от типа данных.	---	0
	Subindex	Подындекс	Подындекс	USINT			
	IsCompleteAccess *1	Полный доступ	Полный доступ	BOOL	Только ЛОЖЬ		ЛОЖЬ

*1. Этот член структуры для данной команды не используется. Всегда записывайте в него значение ЛОЖЬ.

Связанные команды и порядок выполнения

В зависимости от атрибутов данных, записываемых в [интерфейсный модуль EtherCAT; модуль NX, подключенный к интерфейсному модулю EtherCAT; или модуль NX, подключенный к шине NX модуля ЦПУ], эту команду необходимо выполнять вместе с другими командами.

● Порядок действий 1

Для записи данных с указанными ниже атрибутами соблюдайте приведенный ниже порядок действий.

- Атрибут хранения данных при выключенном питании
- Значения обновляются при перезапуске модуля.

1. С помощью команды *NX_ChangeWriteMode* на стр. 2-998 переведите модуль в режим, допускающий запись данных.
2. Используйте команду NX_WriteObj для записи данных в модуль.
3. Используйте команду *NX_SaveParam* на стр. 2-1004 для сохранения записанных данных.
4. С помощью команды *RestartNXUnit* на стр. 2-992 перезапустите модуль.

● Порядок действий 2

Для записи данных с указанными ниже атрибутами соблюдайте приведенный ниже порядок действий.

- Атрибут хранения данных при выключенном питании
- Значения обновляются сразу же после их записи.

1. Используйте команду NX_WriteObj для записи данных в модуль.
2. Используйте команду *NX_SaveParam* на стр. 2-1004 для сохранения записанных данных.

● Порядок действий 3

Для записи данных с указанными ниже атрибутами соблюдайте приведенный ниже порядок действий.

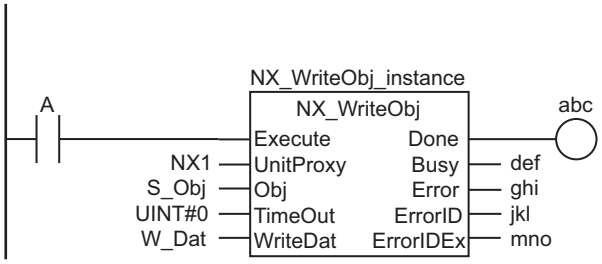
- Без атрибута хранения данных при выключенном питании

1. Используйте команду NX_WriteObj для записи данных в модуль.

Пример записи

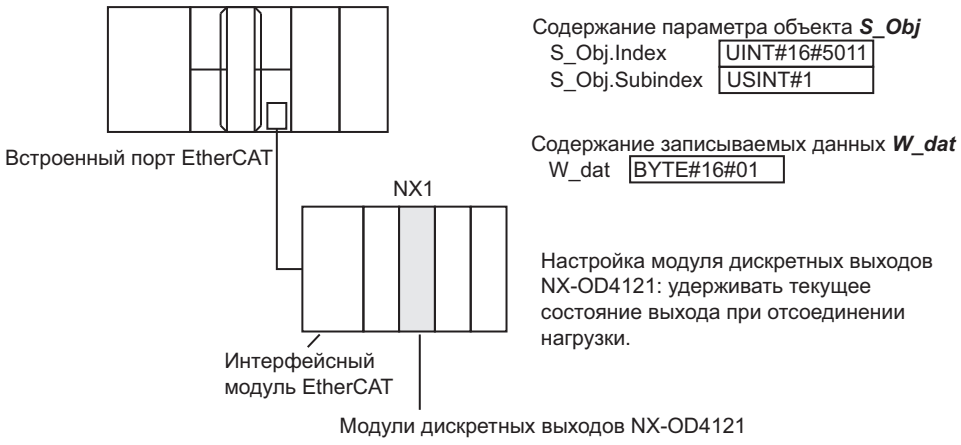
В показанном ниже примере команда используется для того, чтобы настроить модуль дискретных выходов NX-OD4121 так, чтобы в случае отсоединения нагрузки выход удерживался в своем текущем состоянии.
Модулю, в который будут записываться данные, назначена переменная с именем *NX1* и типом данных _sNXUNIT_ID.
Для модуля NX-OD4121 индекс и подындекс параметра Load OFF Output Setting (Настройка работы выхода при отсоединении нагрузки) равны соответственно UINT#16#5011 и USINT#1.
Чтобы удерживалось текущее состояние выхода, в параметр Load Rejection Output Setting записывается значение BYTE#16#01.

LD



ST

```
NX_WriteObj_instance(A, NX1, S_Obj, UINT#0,
                     W_Dat, abc, def, ghi, jkl, mno);
```



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_MBXSlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, способных участвовать в обмене сообщениями	BOOL	Эта переменная указывает возможность связи для каждого ведомого устройства. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_NXB_UnitMsgActiveTbl[i]	Состояние активации обмена сообщениями модуля NX	BOOL	В этой таблице указываются ведомые устройства, которые способны участвовать в обмене сообщениями. С помощью этой переменной можно проверить, возможен ли обмен сообщениями с конкретным ведомым устройством.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если *WriteDat* является массивом, обеспечьте, чтобы общий размер массива совпадал с размером объекта NX для записи в указанном модуле.
- Для *UnitProxy* должна быть указана переменная устройства, которая назначена [интерфейсному модулю EtherCAT; модулю NX, подключенному к интерфейсному модулю EtherCAT; или модулю NX, подключенному к шине NX модуля ЦПУ] на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
  Информацию о назначении переменных устройства см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.
- Всегда используйте переменную в качестве входного параметра, передаваемого в *WriteDat*. При передаче константы произойдет ошибка сборки.
- Чтобы записать и сохранить данные с атрибутом хранения данных при выключенном питании, выполните команду *NX_SaveParam* на стр. 2-1004 после выполнения команды NX_WriteObj. Если модуль будет перезапущен до выполнения команды NX_SaveParam, будут восстановлены предыдущие данные объекта NX.

- Эта команда имеет отношение к ошибкам обмена сообщениями NX. Если одновременно будет выполняться слишком много команд, имеющих отношение к ошибкам обмена сообщениями NX, произойдет ошибка обмена сообщениями NX. Список команд, имеющих отношение к ошибкам обмена сообщениями NX, см. в разделе *A-4 Команды, связанные с ошибками обмена сообщениями NX* на стр. A-38.
- При возникновении ошибки состояние выхода *Error* поменяется на ИСТИНА. В следующей таблице приведены возможные значения переменных *ErrorID* и *ErrorIDEx* и поясняется их значение.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
16#0400	16#00000000	• Значение *UnitProxy* находится за пределами допустимого диапазона. • Значение *TimeOut* находится за пределами допустимого диапазона.
16#0419	16#00000000	• Неправильный тип данных *UnitProxy*. • Неправильный тип данных *WriteDat*.
16#041B	16#00000000	Для *WriteDat* указано более 2 048 байтов данных.
16#2C00	16#00000401	Указанный модуль не поддерживает команду.
	16#00001001 16#00001002 16#00170000 16#00200000 16#00210000	Неправильный входной, выходной или входной-выходной параметр. Проследите, чтобы для входного, выходного или входного-выходного параметра использовался предусмотренный параметр.
	16#00001010	Размер данных указанного объекта NX не согласуется с размером данных, который указан в параметре *WriteDat*.
	16#00001101	Не указан правильный модуль. Проверьте модуль.
	16#0000110B	Размер прочитанных данных слишком велик. Убедитесь, что параметры читаемых данных верны и что читаемые данные указаны верно.
	16#00001110	Не существует объекта, соответствующего значению *Obj.Index*.
	16#00001111	Не существует объекта, соответствующего значению *Obj.Subindex*.
	16#00002101	Указанный объект NX не может быть записан.
	16#00002110	Значение *WriteDat* выходит за диапазон значений, которые могут быть записаны для объекта NX.
	16#00002210	Указанный модуль не находится в режиме, в котором разрешена запись данных.
	16#00002213	Выполнение команды было невозможно, поскольку указанный модуль выполнял проверку ввода-вывода. Выполните команду после завершения проверки ввода-вывода.
	16#00002230	Состояние указанного модуля не согласуется со значением исходного объекта NX для чтения или целевого объекта NX для записи. Если значение *Obj.Index* находится в диапазоне от 0x6000 до 0x6FFF или от 0x7000 до 0x7FFF, выполните указанные ниже действия. • Удалите исходный объект NX для чтения или целевой объект NX для записи из параметров распределения данных ввода-вывода. • Сбросьте ошибку для указанного модуля. • Переведите указанный модуль в режим, в котором не разрешена запись данных.
	16#00002231	Выполнение команды было невозможно, поскольку указанный модуль выполнял инициализацию. Дождитесь, пока модуль начнет работать в обычном режиме, а затем выполните команду.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
	16#0000250F	Не удалось получить доступ к оборудованию. Выполните команду еще раз.
	16#00002601 16#00002602 16#00100000	Указанный модуль не поддерживает эту команду. Проверьте версию модуля.
	16#00002603	Не удалось выполнить команду. Выполните команду еще раз. Убедитесь, что в параметрах выбора используемых каналов активирован (**Enabled**) хотя бы один канал.
	16#00002621	Модуль NX не находится в состоянии, в котором он может подтвердить команду. Подождите некоторое время, а затем выполните команду еще раз.
	16#00010000	Указанного модуля не существует. Убедитесь, что конфигурация модуля верна.
	16#00110000	Указанного номера порта не существует. Убедитесь, что конфигурация модуля верна.
	16#00120000 16#00130000 16#00150000 16#00160000	Неверное значение *UnitProxy*. Снова задайте переменную, которая указывает указанный интерфейсный модуль EtherCAT.
	16#00140000	Указан неправильный адрес узла. Убедитесь, что конфигурация модуля верна.
	16#00300000 16#80010000	Указанный модуль занят. Выполните команду еще раз.
	16#00310000	Для указанного модуля не поддерживается подключение. Проверьте версию модуля.
	16#80000000 16#80050000 16#81010000 16#81020000 16#82020000 16#82030000 16#82060000...16#8FFF0000 16#90010000...16#FFFE0000	Произошла ошибка в сети связи. Выполните команду еще раз.
	16#80020000 16#80030000 16#81030000 16#82000000	Произошла ошибка в сети связи. Уменьшите объем передаваемых данных.
	16#80040000 16#81000000 16#82010000 16#82040000 16#82050000 16#90000000	Произошла ошибка в сети связи. Проверьте модуль и кабельные соединения. Убедитесь, что включен источник питания модуля.
16#2C01	16#00000000	Превышено количество команд, которые могут выполняться одновременно.
16#2C02	16#00000000	Превышено время ожидания при обмене данными.
16#2C03	16#00000000	Неверная длина передаваемого сообщения.

Пример программы

В этом разделе будет рассмотрено два следующих примера, поясняющих работу команды:

- Запись данных в модуль NX с атрибутом Power Off Retain (Атрибут хранения данных при выключенном питании), отражаемых в настройках модуля при перезапуске модуля NX.
- Запись данных в модуль NX с атрибутом Power Off Retain (Атрибут хранения данных при выключенном питании), немедленно отражаемых в настройках модуля.

Пример записи данных, обновляемых при перезапуске модуля

Рассмотрим программу, которая устанавливает параметр объекта **Ch1 Input Moving Average Time** равным 500 мкс для модуля входов переменного тока NX-AD2203, подключенного к интерфейсному модулю EtherCAT.
Адрес узла интерфейсного модуля EtherCAT: 10.

В таблице ниже приведены настройки для параметра объекта **Ch1 Input Moving Average Time**:

Настройка	Значение
Index (Индекс)	16#5004
Subindex (Подындекс)	16#01
Значение для 500 мкс	2

Параметр объекта **Ch1 Input Moving Average Time** имеет атрибут Power OFF Retain (Хранение при выключенном питании), и он обновляется при перезапуске модуля. Поэтому используется приведенная ниже процедура.

1 С помощью команды NX_ChangeWriteMode модуль переводится в режим, допускающий запись данных.

2 Используется команда NX_WriteObj для записи данных в модуль.

3 Используется команда NX_SaveParam для сохранения записанных данных.

4 С помощью команды RestartNXUnit модуль перезапускается.

● Программа на языке LD

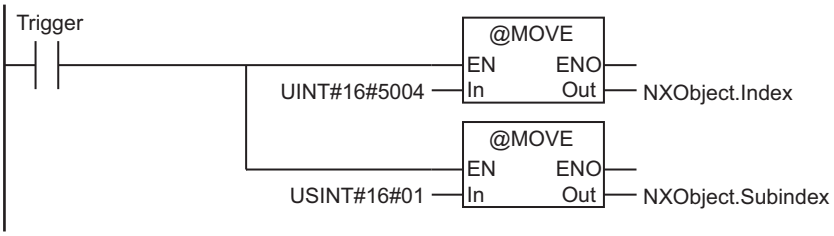
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	ChangeCondition	BOOL	ЛОЖЬ	Условие выполнения для изменения режима записи
	WriteCondition	BOOL	ЛОЖЬ	Условие выполнения для записи данных
	SaveCondition	BOOL	ЛОЖЬ	Условие выполнения для сохранения данных
	RestartCondition	BOOL	ЛОЖЬ	Условие выполнения для перезапуска модуля
	NXUnitProxy	_sNXUNIT_ID		Указание модуля входов постоянного тока
	NXUnitProxy_Coupler	_sNXUNIT_ID		Указание интерфейсного модуля EtherCAT

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	NXObject	_sNXOBJ_ACCESS	(Index:=0, Subindex:=0, IsCompleteAccess:=FALSE)	Параметр объекта
	VarWriteData	UINT	0	Записываемые данные
	NX_ChangeWriteMode_instance	NX_ChangeWriteMode		
	NX_WriteObj_instance	NX_WriteObj		
	NX_SaveParam_instance	NX_SaveParam		
	RestartNXUnit_instance	RestartNXUnit		

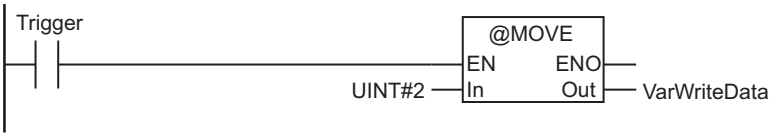
Внешние переменные	Переменная	Константа	Тип данных	Комментарий
	_EC_MBXSlavTbl	☑	ARRAY[1..512] OF BOOL*1	Таблица ведомых устройств, способных участвовать в обмене сообщениями

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

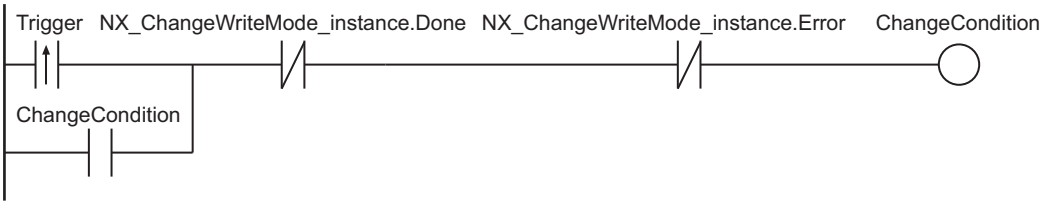
Подготовка параметра объекта.

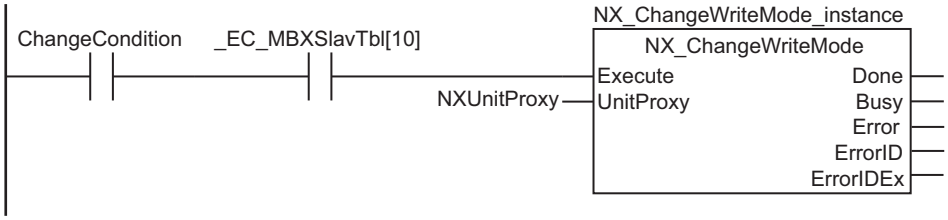


Подготовка записываемых данных.

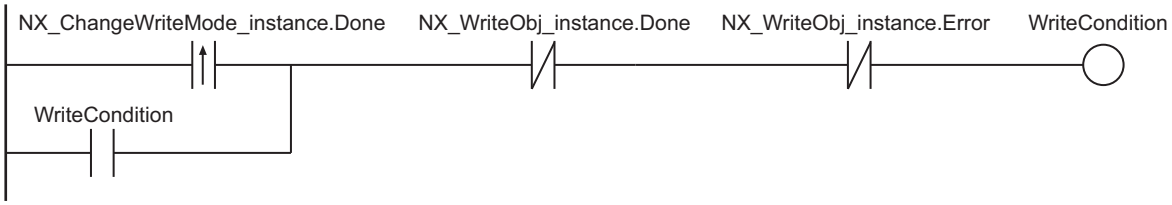


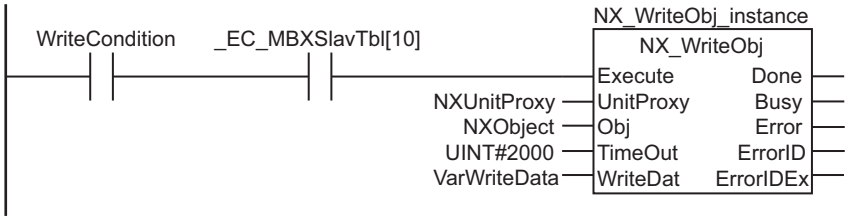
Выполнение команды NX_ChangeWriteMode.



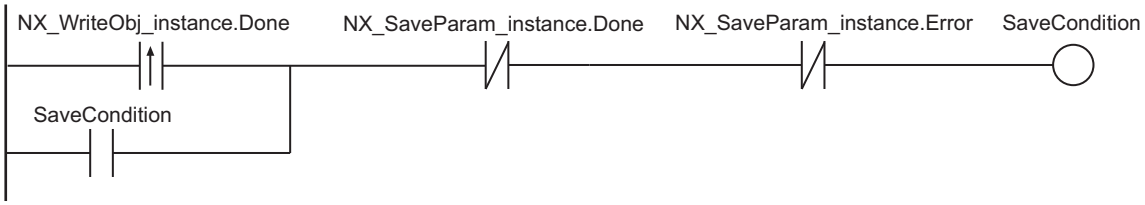


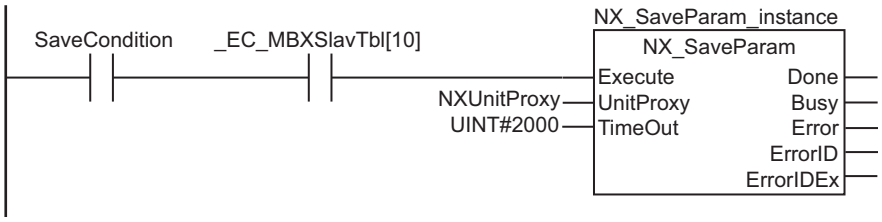
Выполнение команды NX_WriteObj.



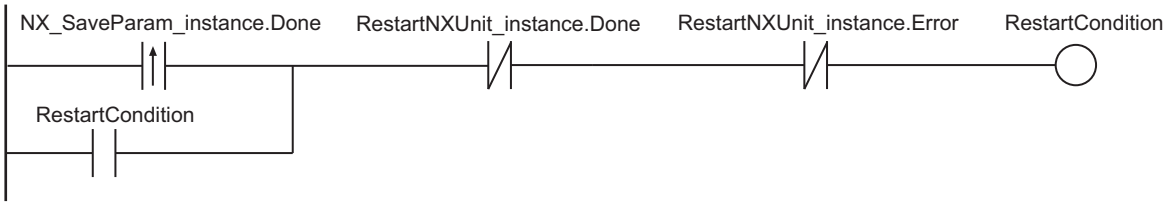


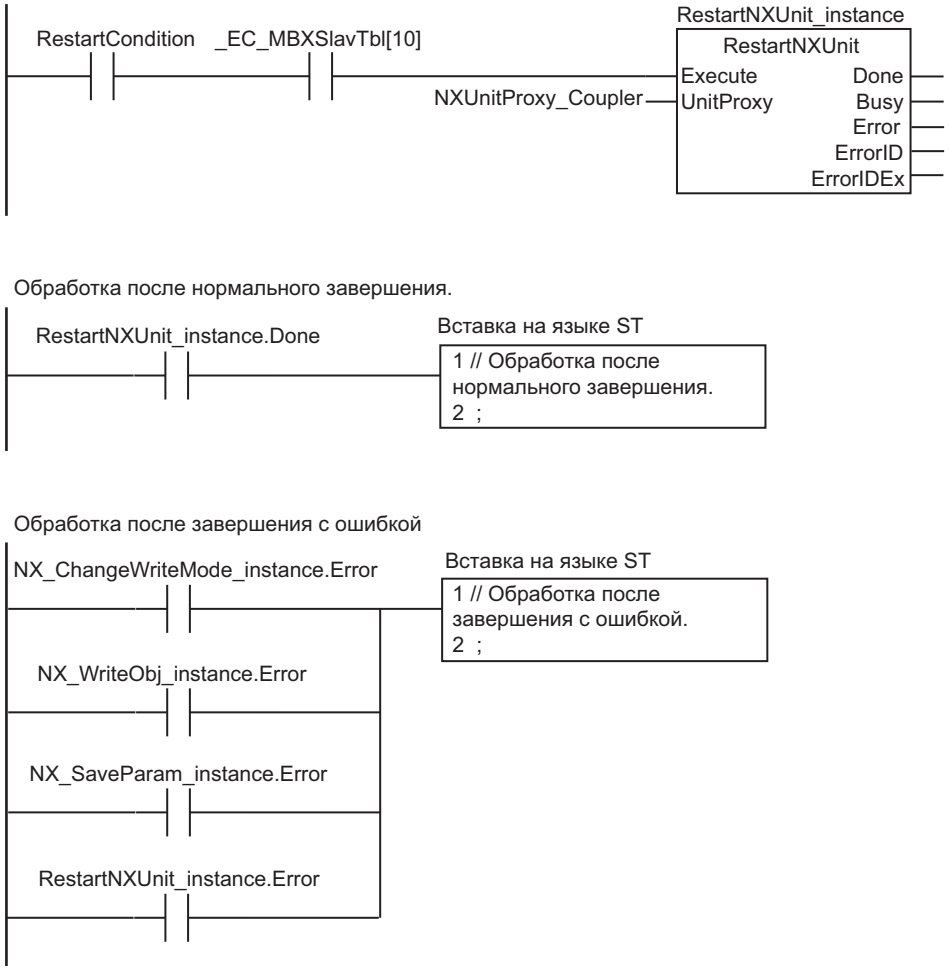
Выполнение команды NX_SaveParam.





Выполнение команды RestartNXUnit.





● Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	ChangeCondition	BOOL	ЛОЖЬ	Условие выполнения для изменения режима записи
	ChangeGo	BOOL	ЛОЖЬ	Выполнение перехода в режим записи
	WriteCondition	BOOL	ЛОЖЬ	Условие выполнения для записи данных
	WriteGo	BOOL	ЛОЖЬ	Выполнение записи данных
	SaveCondition	BOOL	ЛОЖЬ	Условие выполнения для сохранения данных
	SaveGo	BOOL	ЛОЖЬ	Выполнение сохранения данных
	RestartCondition	BOOL	ЛОЖЬ	Условие выполнения для перезапуска модуля
	RestartGo	BOOL	ЛОЖЬ	Выполнение перезапуска модуля
	NXUnitProxy	_sNXUNIT_ID		Указание модуля входов постоянного тока
	NXUnitProxy_Coupler	_sNXUNIT_ID		Указание интерфейсного модуля EtherCAT

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	NXObject	_sNXOBJ_ACCESS	(Index:=0, Subindex:=0, IsCompleteAccess:=FALSE)	Параметр объекта
	VarWriteData	UINT	0	Записываемые данные
	NormalEnd	UINT	0	Нормальное завершение
	ErrorEnd	UINT	0	Завершение с ошибкой
	NX_ChangeWriteMode_instance	NX_ChangeWriteMode		
	NX_WriteObj_instance	NX_WriteObj		
	NX_SaveParam_instance	NX_SaveParam		
	RestartNXUnit_instance	RestartNXUnit		
	R_Trig_instance	R_TRIG		

Внешние переменные	Переменная	Константа	Тип данных	Комментарий
	_EC_MBXSlavTbl	☑	ARRAY[1..512] OF BOOL*1	Таблица ведомых устройств, способных участвовать в обмене сообщениями

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

```
// Подготовка параметра объекта и запись данных.
R_Trig_instance(Clk := Trigger);
IF (R_Trig_instance.Q=TRUE)THEN
  NXObject.Index := UINT#16#5004;
  NXObject.Subindex := USINT#1;
  VarWriteData := UINT#2;
END_IF;

// Выполнение команды NX_ChangeWriteMode.
IF (Trigger = TRUE) THEN
  ChangeCondition := TRUE;
END_IF;

IF ((NX_ChangeWriteMode_instance.Done=TRUE) OR (NX_ChangeWriteMode_instance.Error=T
RUE))THEN
  ChangeCondition := FALSE;
END_IF;

ChangeGo := ChangeCondition & _EC_MBXSlavTbl[10];
NX_ChangeWriteMode_instance(
Execute := ChangeGo,
UnitProxy := NXUnitProxy);
```

```
// Выполнение команды NX_WriteObj.
IF (NX_ChangeWriteMode_instance.Done=TRUE) THEN
  WriteCondition := TRUE;
END_IF;

IF ((NX_WriteObj_instance.Done=TRUE) OR (NX_WriteObj_instance.Error=TRUE))THEN
  WriteCondition := FALSE;
END_IF;

WriteGo := WriteCondition & _EC_MBXSlavTbl[10];
NX_WriteObj_instance(
  Execute := WriteGo,
  UnitProxy := NXUnitProxy,
  Obj := NXObject,
  TimeOut := UINT#2000,
  WriteDat := VarWriteData);

// Выполнение команды NX_SaveParam.
IF (NX_WriteObj_instance.Done=TRUE) THEN
  SaveCondition := TRUE;
END_IF;

IF ((NX_SaveParam_instance.Done=TRUE) OR (NX_SaveParam_instance.Error=TRUE))THEN
  SaveCondition := FALSE;
END_IF;

SaveGo := SaveCondition & _EC_MBXSlavTbl[10];
NX_SaveParam_instance(
  Execute := SaveGo,
  UnitProxy := NXUnitProxy,
  TimeOut := UINT#2000);

// Выполнение команды RestartNXUnit.
IF (NX_SaveParam_instance.Done=TRUE) THEN
  RestartCondition := TRUE;
END_IF;

IF ((RestartNXUnit_instance.Done=TRUE) OR (RestartNXUnit_instance.Error=TRUE)) THEN
  RestartCondition := FALSE;
END_IF;

RestartGo := RestartCondition & _EC_MBXSlavTbl[10];
RestartNXUnit_instance(
  Execute := SaveGo,
  UnitProxy := NXUnitProxy_Coupler);

IF (RestartNXUnit_instance.Done=TRUE) THEN
  // Обработка после нормального завершения.
  NormalEnd := NormalEnd + UINT#1;
ELSIF ((NX_ChangeWriteMode_instance.Error=TRUE) OR (NX_WriteObj_instance.Error=TRUE
)
```

```
  OR (NX_SaveParam_instance.Error=TRUE) OR (RestartNXUnit_instance.Error=TRUE)) THE
N
  // Обработка после завершения с ошибкой.
  ErrorEnd := ErrorEnd + UINT#1;
END_IF;
```

Пример записи данных, обновляемых немедленно

Рассмотрим программу, которая устанавливает параметр объекта **Ch1 Offset Value (One-point Correction)** равным 0,3 °C для модуля температурных входов NX-TS2101, подключенного к интерфейсному модулю EtherCAT.
Адрес узла интерфейсного модуля EtherCAT: 10.

В таблице ниже приведены настройки для параметра объекта **Ch1 Offset Value (One-point Correction)**:

Настройка	Значение
Index (Индекс)	16#5010
Subindex (Подындекс)	16#01
Записываемое значение	0,3

Параметр объекта **Ch1 Offset Value (One-point Correction)** имеет атрибут Power OFF Retain (Хранение при выключенном питании), и он обновляется сразу после записи данных. Поэтому используется приведенная ниже процедура.

1 Используется команда NX_WriteObj для записи данных в модуль.

2 Используется команда NX_SaveParam для сохранения записанных данных.

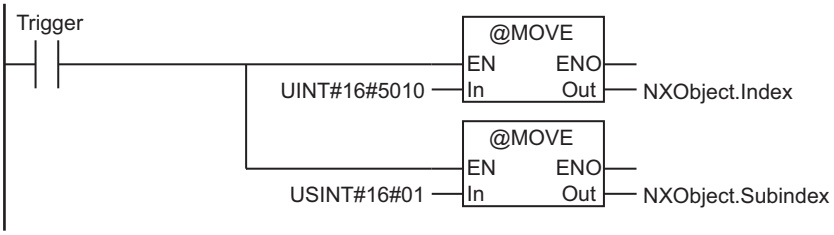
● Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	WriteCondition	BOOL	ЛОЖЬ	Условие выполнения для записи данных
	SaveCondition	BOOL	ЛОЖЬ	Условие выполнения для сохранения данных
	NXUnitProxy	_sNXUNIT_ID		Указание модуля входов переменного тока
	NXUnitProxy_Coupler	_sNXUNIT_ID		Указание интерфейсного модуля EtherCAT
	NXObject	_sNXOBJ_ACCESS	(Index:=0, Subindex:=0, IsCompleteAccess:=FALSE)	Параметр объекта
	VarWriteData	Вещественный	0,0	Записываемые данные
	NX_WriteObj_instance	NX_WriteObj		
	NX_SaveParam_instance	NX_SaveParam		

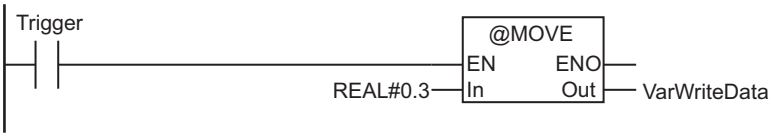
Внешние переменные	Переменная	Константа	Тип данных	Комментарий
	_EC_MBXSlavTbl	☑	ARRAY[1..512] OF BOOL*1	Таблица ведомых устройств, способных участвовать в обмене сообщениями

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

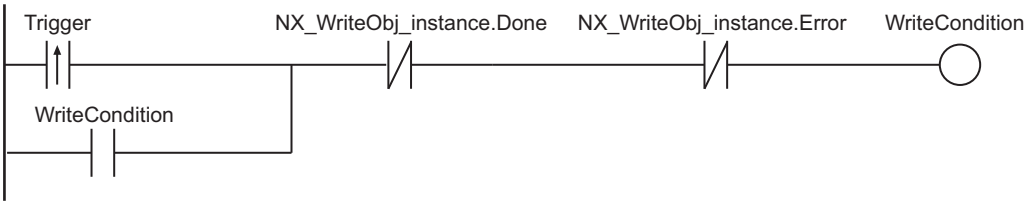
Подготовка параметра объекта.

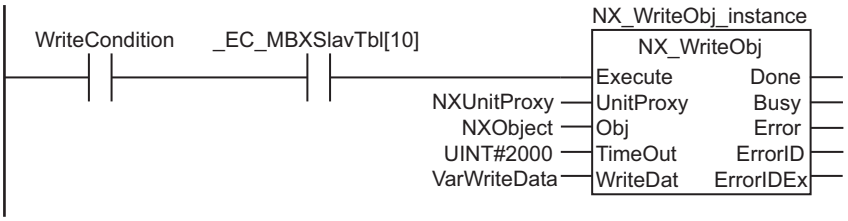


Подготовка записываемых данных.

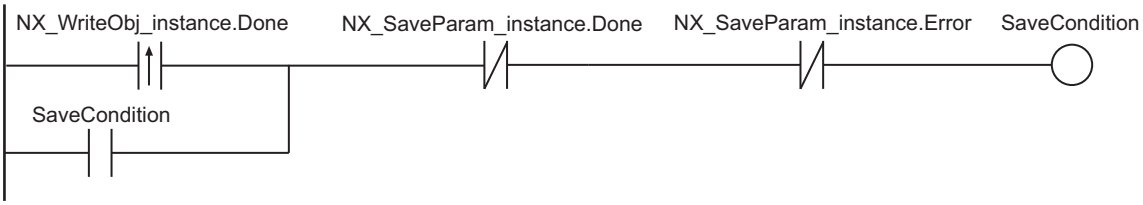


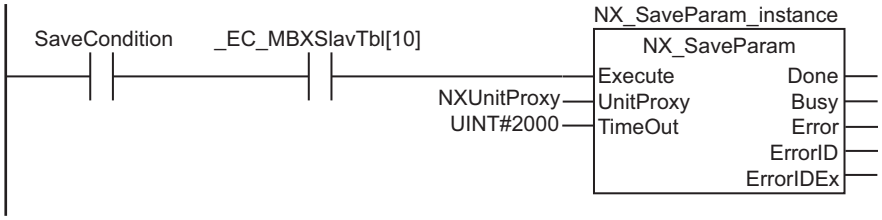
Выполнение команды NX_WriteObj.



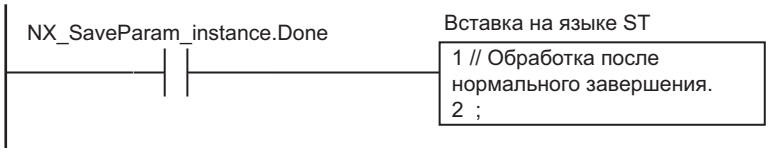


Выполнение команды NX_SaveParam.

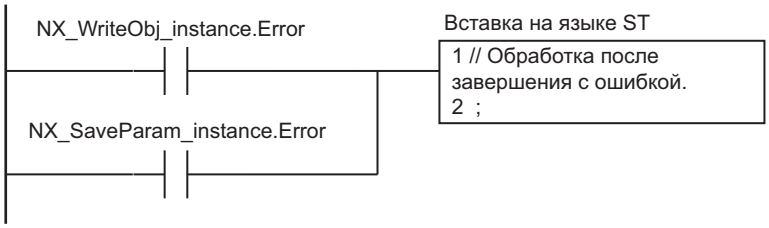




Обработка после нормального завершения.



Обработка после завершения с ошибкой



● Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	WriteCondition	BOOL	ЛОЖЬ	Условие выполнения для записи данных
	WriteGo	BOOL	ЛОЖЬ	Выполнение записи данных
	SaveCondition	BOOL	ЛОЖЬ	Условие выполнения для сохранения данных
	SaveGo	BOOL	ЛОЖЬ	Выполнение сохранения данных
	NXUnitProxy	_sNXUNIT_ID		Указание модуля температурных входов
	NXUnitProxy_Coupler	_sNXUNIT_ID		Указание интерфейсного модуля EtherCAT
	NXObject	_sNXOBJ_ACCESS	(Index:=0, Subindex:=0, IsCompleteAccess:=FALSE)	Параметр объекта
	VarWriteData	REAL	0,0	Записываемые данные
	NormalEnd	UINT	0	Нормальное завершение
	ErrorEnd	UINT	0	Завершение с ошибкой
	NX_WriteObj_instance	NX_WriteObj		
	NX_SaveParam_instance	NX_SaveParam		
	R_Trig_Instance	R_TRIG		

Внешние переменные	Переменная	Константа	Тип данных	Комментарий
	_EC_MBXSlavTbl	☑	ARRAY[1..512] OF BOOL*1	Таблица ведомых устройств, способных участвовать в обмене сообщениями

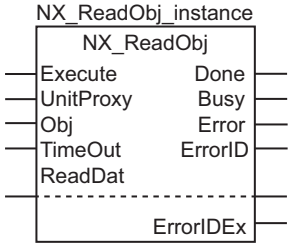
*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

```
// Подготовка параметра объекта и запись данных.
R_Trig_instance(Clk := Trigger);
IF (R_Trig_instance.Q=TRUE)THEN
  NXObject.Index := UINT#16#5004;
  NXObject.Subindex := USINT#1;
  VarWriteData := UINT#2;
END_IF;

// Выполнение команды NX_WriteObj.
IF (Trigger=TRUE) THEN
  WriteCondition := TRUE;
END_IF;

IF ((NX_WriteObj_instance.Done=TRUE) OR (NX_WriteObj_instance.Error=TRUE))THEN
  WriteCondition := FALSE;
END_IF;

WriteGo := WriteCondition & _EC_MBXSlavTbl[10];
NX_WriteObj_instance(
  Execute := WriteGo,
  UnitProxy := NXUnitProxy,
  Obj := NXObject,
  TimeOut := UINT#2000,
  WriteDat := VarWriteData);

// Выполнение команды NX_SaveParam.
IF (NX_WriteObj_instance.Done=TRUE) THEN
  SaveCondition := TRUE;
END_IF;

IF ((NX_SaveParam_instance.Done=TRUE) OR (NX_SaveParam_instance.Error=TRUE))THEN
  SaveCondition := FALSE;
END_IF;

SaveGo := SaveCondition & _EC_MBXSlavTbl[10];
NX_SaveParam_instance(
  Execute := SaveGo,
  UnitProxy := NXUnitProxy,
  TimeOut := UINT#2000);

IF (NX_SaveParam_instance.Done=TRUE) THEN
```

```
  // Обработка после нормального завершения.
  NormalEnd := NormalEnd + UINT#1;
ELSIF ((NX_WriteObj_instance.Error=TRUE) OR (NX_SaveParam_instance.Error=TRUE)) THE
N
  // Обработка после завершения с ошибкой.
  ErrorEnd := ErrorEnd + UINT#1;
END_IF;
```

NX_ReadObj

Команда NX_ReadObj производит чтение данных из объекта NX в интерфейсном модуле EtherCAT или модуле NX.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_ReadObj	Чтение объекта модуля NX	FB		NX_ReadObj_instance(Execute, UnitProxy, Obj, TimeOut, ReadDat, Done, Busy, Error, ErrorID, ErrorIDEx);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.05 или более поздней и Sysmac Studio версии 1.06 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitProxy	Указанный модуль	Вход	Модуль, из которого нужно прочитать данные	---	---	*1
Obj	Параметр объекта		Параметр объекта			---
TimeOut	Время ожидания		Время ожидания Если задано значение 0, время ожидания составляет 2,0 с.	0...60 000	мс	2000 (2,0 с)
ReadDat	Прочитанные данные	Вход-выход	Данные, прочитанные из объекта NX	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitProxy	Подробные сведения о структуре _sNXUNIT_ID см. в разделе *Функция* на стр. 2-1132.																			
Obj	Подробные сведения о структуре _sNXOBJ_ACCESS см. в разделе *Функция* на стр. 2-1132.																			
TimeOut							OK													
ReadDat	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать массив.																			

Функция

Команда NX_ReadObj производит чтение данных из объекта NX в [интерфейсном модуле EtherCAT; модуле NX, подключенном к интерфейсному модулю EtherCAT; или модуле NX, подключенном к шине NX модуля ЦПУ], и сохраняет прочитанные данные в переменную *ReadDat*. Модуль, из которого должны быть прочитаны данные, указывается с помощью параметра *UnitProxy*.

Параметр *TimeOut* задает время ожидания. Если ответ не возвращается в течение времени ожидания, считается, что произошел сбой связи. В этом случае данные не считываются.

Для переменной *UnitProxy* используется структурный тип данных _sNXUNIT_ID. Значения членов структуры приведены в таблице ниже.

Имя		Значение	Содержание	Тип данных
UnitProxy		Указанный модуль	Указанный модуль	_sNXUNIT_ID
	NodeAdr	Адрес узла	Адрес узла интерфейсного модуля	UINT
	IPAdr	IP-адрес	IP-адрес интерфейсного модуля	BYTE[5]
	UnitNo	Номер модуля	Номер модуля указанного модуля	UDINT
	Path	Путь	Информация о пути к указанному модулю	BYTE[64]
	PathLength	Допустимая длина пути *Path*	Допустимая длина пути *Path*	USINT

В *UnitProxy* следует передать переменную устройства, которая назначена указанному модулю.

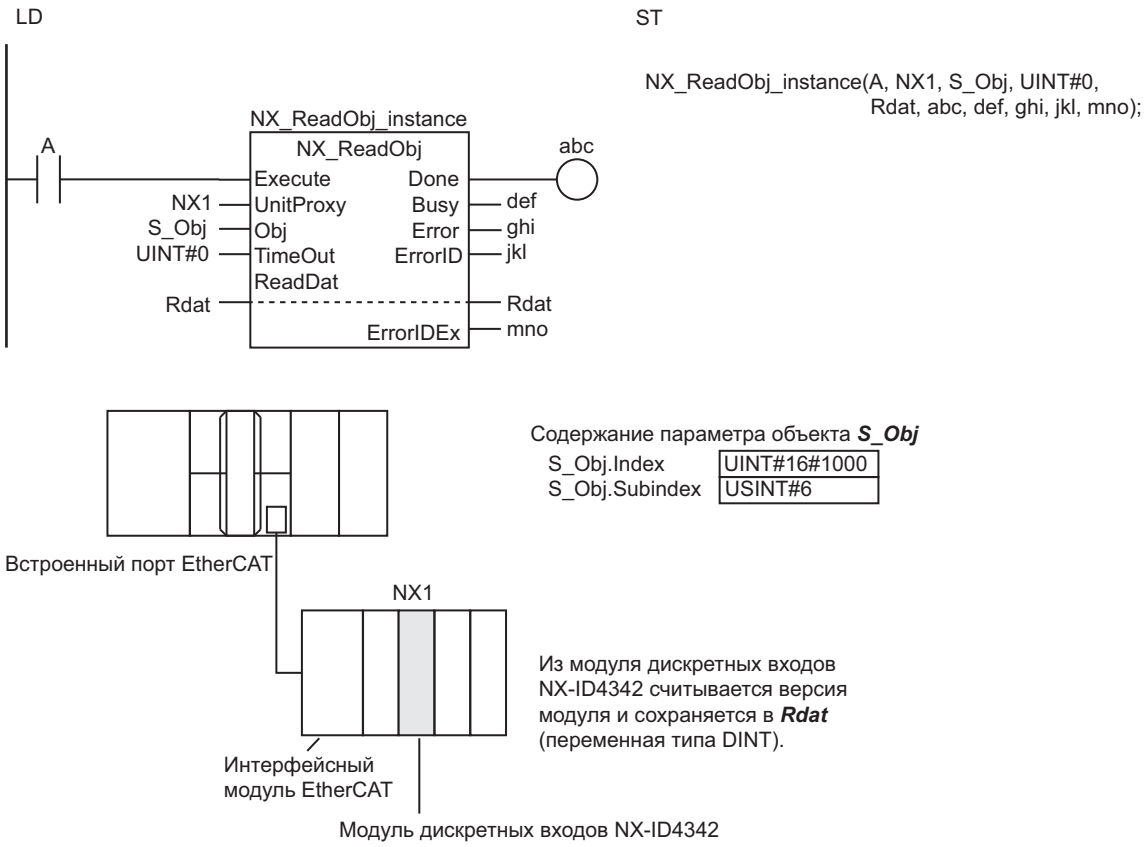
Для переменной *Obj* используется структурный тип данных _sNXOBJ_ACCESS. Значения членов структуры приведены в таблице ниже.

Имя		Значение	Содержание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Obj		Параметр объекта	Параметр объекта	_sNXOBJ_ACCESS	---	---	---
	Index	Индекс	Индекс	UINT	Зависит от типа данных.	---	0
	Subindex	Подындекс	Подындекс	USINT			
	IsCompleteAccess *1	Полный доступ	Полный доступ	BOOL	Только ЛОЖЬ		ЛОЖЬ

*1. Этот член структуры для данной команды не используется. Всегда записывайте в него значение ЛОЖЬ.

Пример записи

Ниже показан пример использования команды для чтения версии модуля из модуля дискретных входов NX-ID4342.
Прочитанные данные сохраняются в переменную *Rdat* типа UDINT.
Модулю, из которого будут считываться данные, назначена переменная с именем *NX1* и типом данных _sNXUNIT_ID.
Для модуля NX-ID4342 индекс и подындекс параметра Unit version (Версия модуля) равны соответственно UINT#16#1000 и USINT#6.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_MBXSlavTbl[i] "i" — это адрес узла.	Таблица ведомых устройств, способных участвовать в обмене сообщениями	BOOL	Эта переменная указывает возможность связи для каждого ведомого устройства. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_NXB_UnitMsgActiveTbl[i]	Состояние активации обмена сообщениями модуля NX	BOOL	В этой таблице указываются ведомые устройства, которые способны участвовать в обмене сообщениями. С помощью этой переменной можно проверить, возможен ли обмен сообщениями с конкретным ведомым устройством.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если *ReadDat* является массивом, обеспечьте, чтобы общий размер массива совпадал с размером объекта NX для чтения в указанном модуле.

- Для *UnitProxy* должна быть указана переменная устройства, которая назначена [интерфейсному модулю EtherCAT; модулю NX, подключенному к интерфейсному модулю EtherCAT; или модулю NX, подключенному к шине NX модуля ЦПУ] на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
  Информацию о назначении переменных устройства см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.
- Эта команда имеет отношение к ошибкам обмена сообщениями NX. Если одновременно будет выполняться слишком много команд, имеющих отношение к ошибкам обмена сообщениями NX, произойдет ошибка обмена сообщениями NX. Список команд, имеющих отношение к ошибкам обмена сообщениями NX, см. в разделе *A-4 Команды, связанные с ошибками обмена сообщениями NX* на стр. A-38.
- При возникновении ошибки состояние выхода *Error* меняется на «ИСТИНА». В следующей таблице приведены возможные значения переменных *ErrorID* и *ErrorIDEx* и поясняется их значение.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
16#0400	16#00000000	• Значение *UnitProxy* находится за пределами допустимого диапазона. • Значение *TimeOut* находится за пределами допустимого диапазона.
16#0410	16#00000000	*ReadDat* содержит строковое значение (STRING) и не завершается символом NULL.
16#0419	16#00000000	• Неправильный тип данных *UnitProxy*. • Неправильный тип данных *ReadDat*.
16#041C	16#00000000	Размер *ReadDat* не совпадает с размером читаемого объекта NX.
16#2C00	16#00000401	Указанный модуль не поддерживает команду.
	16#00001001 16#00001002 16#00170000 16#00200000 16#00210000	Неправильный входной, выходной или входной-выходной параметр. Проследите, чтобы для входного, выходного или входного-выходного параметра использовался предусмотренный параметр.
	16#00001010	Размер данных указанного объекта NX не согласуется с размером данных, который указан в параметре *WriteDat*.
	16#00001101	Не указан правильный модуль. Проверьте модуль.
	16#0000110B	Размер прочитанных данных слишком велик. Убедитесь, что параметры читаемых данных верны и что читаемые данные указаны верно.
	16#00001110	Не существует объекта, соответствующего значению *Obj.Index*.
	16#00001111	Не существует объекта, соответствующего значению *Obj.Subindex*.
	16#00002101	Указанный объект NX не может быть записан.
	16#00002110	Значение *WriteDat* выходит за диапазон значений, которые могут быть записаны для объекта NX.
	16#00002210	Указанный модуль не находится в режиме, в котором разрешена запись данных.
	16#00002213	Выполнение команды было невозможно, поскольку указанный модуль выполнял проверку ввода-вывода. Выполните команду после завершения проверки ввода-вывода.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
	16#00002230	Состояние указанного модуля не согласуется со значением исходного объекта NX для чтения или целевого объекта NX для записи. Если значение *Obj.Index* находится в диапазоне от 0x6000 до 0x6FFF или от 0x7000 до 0x7FFF, выполните указанные ниже действия. • Удалите исходный объект NX для чтения или целевой объект NX для записи из параметров распределения данных ввода-вывода. • Сбросьте ошибку для указанного модуля. • Переведите указанный модуль в режим, в котором не разрешена запись данных.
	16#00002231	Выполнение команды было невозможно, поскольку указанный модуль выполнял инициализацию. Дождитесь, пока модуль начнет работать в обычном режиме, а затем выполните команду.
	16#0000250F	Не удалось получить доступ к оборудованию. Выполните команду еще раз.
	16#00002601 16#00002602 16#00100000	Указанный модуль не поддерживает эту команду. Проверьте версию модуля.
	16#00002603	Не удалось выполнить команду. Выполните команду еще раз. Убедитесь, что в параметрах выбора используемых каналов активирован (**Enabled**) хотя бы один канал.
	16#00002621	Модуль NX не находится в состоянии, в котором он может подтвердить команду. Подождите некоторое время, а затем выполните команду еще раз.
	16#00010000	Указанного модуля не существует. Убедитесь, что конфигурация модуля верна.
	16#00110000	Указанного номера порта не существует. Убедитесь, что конфигурация модуля верна.
	16#00120000 16#00130000 16#00150000 16#00160000	Неверное значение *UnitProxy*. Снова задайте переменную, которая указывает указанный интерфейсный модуль EtherCAT.
	16#00140000	Указан неправильный адрес узла. Убедитесь, что конфигурация модуля верна.
	16#00300000 16#80010000	Указанный модуль занят. Выполните команду еще раз.
	16#00310000	Для указанного модуля не поддерживается подключение. Проверьте версию модуля.
	16#80000000 16#80050000 16#81010000 16#81020000 16#82020000 16#82030000 16#82060000...16#8FFF0000 16#90010000...16#FFFE0000	Произошла ошибка в сети связи. Выполните команду еще раз.
	16#80020000 16#80030000 16#81030000 16#82000000	Произошла ошибка в сети связи. Уменьшите объем передаваемых данных.

Значение *ErrorID*	Значение *ErrorIDEx*	Значение
	16#80040000 16#81000000 16#82010000 16#82040000 16#82050000 16#90000000	Произошла ошибка в сети связи. Проверьте модуль и кабельные соединения. Убедитесь, что включен источник питания модуля.
16#2C01	16#00000000	Превышено количество команд, которые могут выполняться одновременно.
16#2C02	16#00000000	Превышено время ожидания при обмене данными.

Пример программы

В данном примере выполняется чтение значения параметра объекта *I/O Refresh Method 1(Способ обновления вх./вых. 1)* из интерфейсного модуля EtherCAT NX-ECC201.
Адрес узла интерфейсного модуля EtherCAT: 10.

В следующей таблице приведены значения индекса и подындекса параметра объекта *I/O Refresh Method 1*.

Настройка	Значение
Index (Индекс)	16#4002
Subindex (Подындекс)	16#01

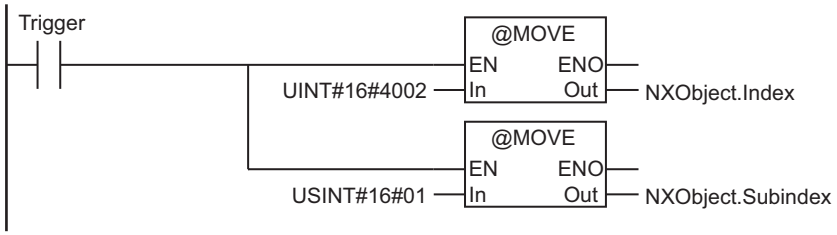
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	ReadCondition	BOOL	ЛОЖЬ	Условие выполнения для чтения данных
	NXUnitProxy	_sNXUNIT_ID		Указание модуля
	NXObject	_sNXOBJ_ACCESS	(Index:=0, Subindex:=0, IsCompleteAccess:=FALSE)	Параметр объекта
	IoRefreshMethod	USINT	0	Прочитанные данные
	NX_ReadObj_instance	NX_ReadObj		

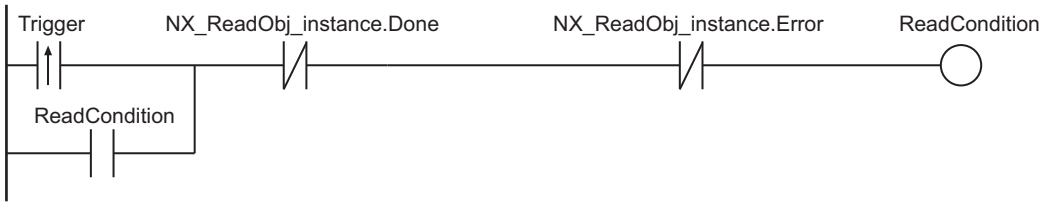
Внешние переменные	Переменная	Константа	Тип данных	Комментарий
	_EC_MBXSlavTbl	☑	ARRAY[1..512] OF BOOL*1	Таблица ведомых устройств, способных участвовать в обмене сообщениями

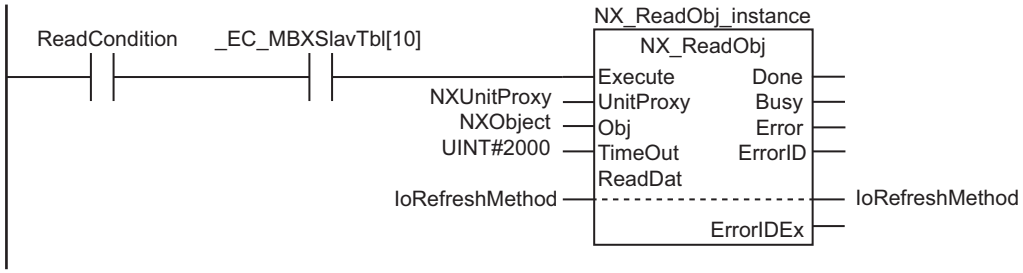
*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

Подготовка параметра объекта.

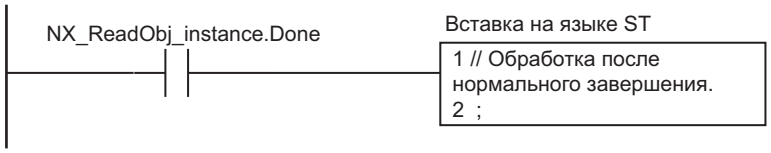


Выполнение команды NX_ReadObj.

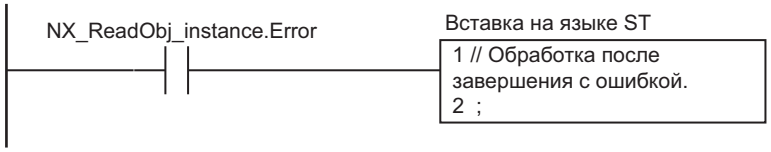




Обработка после нормального завершения.



Обработка после завершения с ошибкой.



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	ReadCondition	BOOL	ЛОЖЬ	Условие выполнения для чтения данных
	ReadGo	BOOL	ЛОЖЬ	Выполнение чтения данных
	NXUnitProxy	_sNXUNIT_ID		Указание модуля

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	NXObject	_sNXOBJ_ACCESS	(Index:=0, Subindex:=0, IsCompleteAccess:=FALSE)	Параметр объекта
	IoRefreshMethod	USINT	0	Прочитанные данные
	NormalEnd	UINT	0	Нормальное завершение
	ErrorEnd	UINT	0	Завершение с ошибкой
	R_Trig_instance	R_Trig		
	NX_ReadObj_instance	NX_ReadObj		

Внешние переменные	Переменная	Константа	Тип данных	Комментарий
	_EC_MBXSlavTbl	☑	ARRAY[1..512] OF BOOL*1	Таблица ведомых устройств, способных участвовать в обмене сообщениями

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

```
// Подготовка параметра объекта.
R_Trig_instance(Clk := Trigger);
IF (R_Trig_instance.Q=TRUE)THEN
  NXObject.Index := UINT#16#4002;
  NXObject.Subindex := USINT#1;
END_IF;

// Выполнение команды NX_ReadObj.
IF (Trigger=TRUE) THEN
  ReadCondition := TRUE;
END_IF;

IF ( (NX_ReadObj_instance.Done=TRUE) OR (NX_ReadObj_instance.Error=TRUE) ) THEN
  ReadCondition := FALSE;
END_IF;

ReadGo := ReadCondition & _EC_MBXSlavTbl[10];
NX_ReadObj_instance(
  Execute := ReadGo,
  UnitProxy := NXUnitProxy,
  Obj := NXObject,
  TimeOut := UINT#2000,
  ReadDat := IoRefreshMethod);

// Обработка после выполнения команды.
IF (NX_ReadObj_instance.Done=TRUE) THEN
  // Обработка после нормального завершения.
  NormalEnd := NormalEnd + UINT#1;
ELSIF (NX_ReadObj_instance.Error=TRUE) THEN
  // Обработка после завершения с ошибкой.
```

```
  ErrorEnd := ErrorEnd + UINT#1;
END_IF;
```

Команды для обмена данными по интерфейсу IO-Link

Команда	Имя	Стр.
IOL_ReadObj	Чтение из объекта устройства IO-Link	стр. 2-1142
IOL_WriteObj	Запись в объект устройства IO-Link	стр. 2-1151

IOL_ReadObj

Команда IOL_ReadObj производит чтение данных из объектов устройств IO-Link.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
IOL_ReadObj	Чтение из объекта устройства IO-Link	FB	IOL_ReadObj_instance IOL_ReadObj Execute — Done DevicePort — Busy DeviceObj — Error RetryCfg — ErrorID ReadDat ErrorType ReadSize	IOL_ReadObj_instance(Execute, DevicePort, DeviceObj, RetryCfg, ReadDat, Done, Busy, Error, ErrorID, ErrorType, ReadSize);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.12 или более поздней и Sysmac Studio версии 1.16 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
DeviceObj	Параметр объекта устройства IO-Link		Указание объекта устройства IO-Link	---	---	---
RetryCfg	Настройка повтора выполнения		Настройка повторных попыток выполнения команды	---	---	---
ReadDat	Прочитанные данные	Вход-выход	Данные, прочитанные из устройства IO-Link	Зависит от типа данных.	---	0
ErrorType	Тип ошибки	Выход	Код ошибки, возвращаемый устройством IO-Link, сохраняется, когда *ErrorID = 4800 hex*.	16#0000...16#FFFF	---	---
ReadSize	Объем прочитанных данных		Объем данных, сохраненных в *ReadDat*	10#1...10#232	Байты	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1143.																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DeviceObj	Подробные сведения о структуре _sIOLOBJ_ACCESS см. в разделе *Функция* на стр. 2-1143.																			
RetryCfg	Подробные сведения о структуре _sIOL_RETRY_CFG см. в разделе *Функция* на стр. 2-1143.																			
ReadDat	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также можно указать массив.																			
ErrorType			ОК																	
ReadSize							ОК													

Функция

Команда IOL_ReadObj производит чтение данных объектов из устройств IO-Link.

Во входной переменной *DevicePort* следует указать модуль ведущего устройства IO-Link и номер порта, к которому подключено целевое устройство IO-Link, из которого считываются данные.
Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Резерв	Резерв	Резерв	---	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2 3: порт 3 4: порт 4 5: порт 5 6: порт 6 7: порт 7 8: порт 8	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства. Укажите значение _DeviceNXUnit для модуля ведущего устройства IO-Link типа NX или значение _DeviceEcatSlave для модуля ведущего устройства IO-Link типа GX.

Переменная, используемая для указания устройства, определяется указанным типом устройства.

Для данной команды она определяется следующим образом:
Для указания устройства типа NX используется переменная *NxUnit*. Переменная *EcatSlave* в данном случае не используется.
В *NxUnit* следует передать переменную устройства, которая назначена указываемому устройству.
Для указания устройства типа GX используется переменная *EcatSlave*. Переменная *NxUnit* в данном случае не используется.
В *EcatSlave* следует передать переменную устройства, которая назначена указываемому устройству.

Используйте переменную *PortNo* для указания номера порта, к которому подключено устройство IO-Link.
Количество портов зависит от типа модуля ведущего устройства IO-Link.
Модуль типа NX: 1…4
Модуль типа GX: 1…8

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.

Используйте входную переменную *DeviceObj* для указания параметра объекта для устройства IO-Link, из которого читаются данные.
Для входной переменной *DeviceObj* используется структурный тип данных _sIOLOBJ_ACCESS. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DeviceObj	Параметр объекта устройства IO-Link	Указание объекта устройства IO-Link	_sIOLOBJ_ACCESS	---	---	---
Index	Индекс	Индекс	UINT	Зависит от типа данных.	---	---
Subindex	Подындекс	Задайте значение *0* для чтения из всего индекса.	USINT	Зависит от типа данных.	---	---

Используйте входную переменную *RetryCfg* для настройки повторных попыток выполнения команды.
Для переменной *RetryCfg* используется структурный тип данных _sIOL_RETRY_CFG. Описание приведено в таблице ниже.

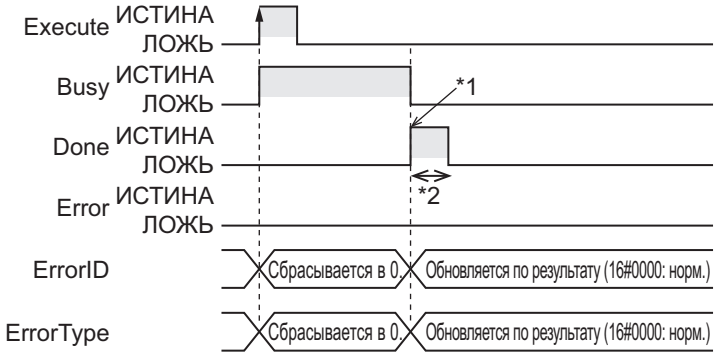
Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
RetryCfg	Настройка повтора выполнения	Настройка повторных попыток выполнения команды	_sIOL_RETRY_CFG	---	---	---
TimeOut	Время ожидания	Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	TIME	0...300 с	---	T#2.0s
RetryNum	Количество повторных попыток	Количество повторных попыток при превышении времени ожидания Если задано значение *0*, предпринимается 3 повторных попытки.	UINT	Зависит от типа данных.	Кол-во раз	3

Данные, прочитанные из устройства IO-Link, сохраняются во входную-выходную переменную *ReadDat*.

Временные диаграммы

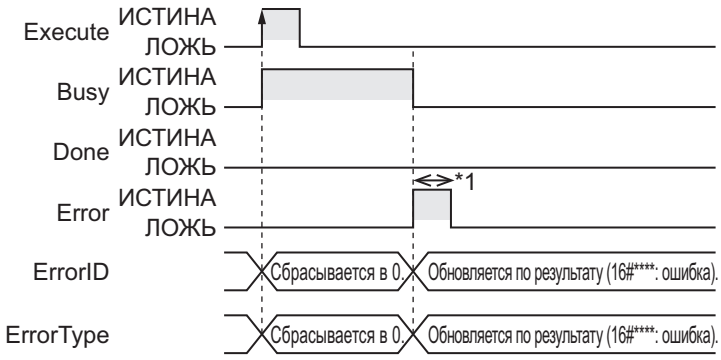
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Чтение завершено.
*2. Период выполнения задачи

● Завершение с ошибкой



*1. Период выполнения задачи

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_MBXSlavTbl	Таблица ведомых устройств, способных участвовать в обмене сообщениями	ARRAY[1..512] OF BOOL*1	В этой таблице указываются ведомые устройства, которые способны участвовать в обмене сообщениями. Таблица содержит данные для ведомых устройств в порядке возрастания адресов узлов. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: ARRAY [1..192] OF BOOL.

Меры предосторожности для обеспечения надлежащей эксплуатации

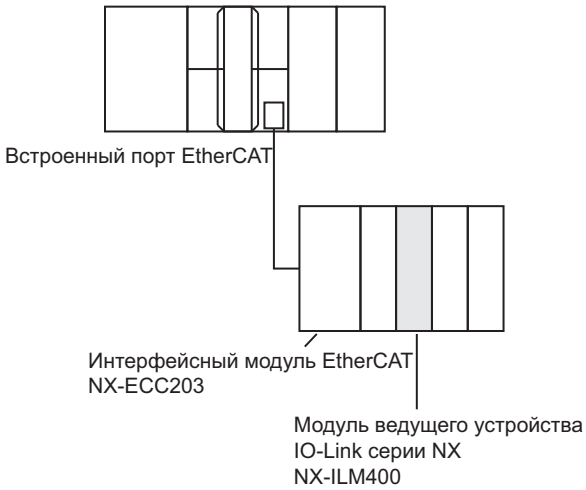
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Для параметров *DevicePort.NxUnit* и *DevicePort.EcatSlave* должна быть указана соответствующая переменная устройства, которая назначена модулю ведущего устройства IO-Link на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio. Информацию о назначении переменных устройства см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.
- Размер переменной, указываемой для *ReadDat*, должен быть больше размера фактически считываемого объекта.
- Если параметр *ReadDat* имеет строковый тип (STRING), размер указываемой переменной должен быть равен размеру фактически считываемой строки с учетом добавления пустого символа (NULL).
- Если параметр *ReadDat* имеет строковый тип (STRING), в переменную *ReadSize* выдается размер без учета пустого символа (NULL).
- Всегда используйте переменную в качестве входного параметра, передаваемого в *ReadDat*. При передаче константы произойдет ошибка сборки.
- Для модуля ведущего устройства IO-Link, независимо от его типа (NX или GX), можно выполнять только одну команду за раз.
- Эту команду невозможно использовать в событийной задаче. В этом случае произойдет ошибка компиляции.
- Эта команда выполняется, когда вход *Execute* переходит в состояние ИСТИНА. Команда не выполняется, если вход *Execute* постоянно находится в состоянии ИСТИНА.
- Для команд IOL_ReadObj и IOL_WriteObj можно определить максимум 64 экземпляра.
- В указанных ниже случаях произойдет ошибка.
  a) Для параметра *DevicePort.NxUnit* или *DevicePort.EcatSlave* указано значение, выходящее за пределы допустимого диапазона.
  b) Размер считываемого объекта устройства IO-Link больше, чем размер переменной *ReadDat*. Если возникает эта ошибка, прочитанные данные в переменную *ReadDat* не сохраняются.
  c) От устройства IO-Link получен ответ с кодом ошибки.
  Старшие восемь битов представляют код ошибки (*ErrorCode*), а младшие восемь битов представляют дополнительный код (*AdditionalCode*).

Сведения о значениях *ErrorCode* и *AdditionalCode* см. в описании типов ошибок в документе «IO-Link Communication Specification» («Техническое описание интерфейса связи IO-Link»). Описание типов ошибок можно найти на веб-сайте консорциума IO-Link: http://www.io-link.com/

d) Указанного модуля ведущего устройства IO-Link не существует.
e) Превышено максимальное количество сообщений, которые может обработать ведущее устройство IO-Link. Команда не может быть выполнена, так как ведущее устройство IO-Link обрабатывает сообщения от других приложений.
f) Указанный модуль ведущего устройства IO-Link не находится в состоянии, в котором он может принимать сообщения.
g) Было выполнено более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
h) Превышено время ожидания при обмене данными.
i) Указанный порт модуля ведущего устройства IO-Link не находится в режиме IO-Link. Порт отключен или находится в режиме SIO.
j) Устройство IO-Link не подключено к указанному порту модуля ведущего устройства IO-Link.
k) На указанный порт модуля ведущего устройства IO-Link не подается питание ввода-вывода.
l) В указанном порте модуля ведущего устройства IO-Link произошла ошибка проверки или ошибка связи.

Пример программы

В данном примере модуль ведущего устройства IO-Link (NX-ILM400) подключен к интерфейсному модулю EtherCAT (NX-ECC203).



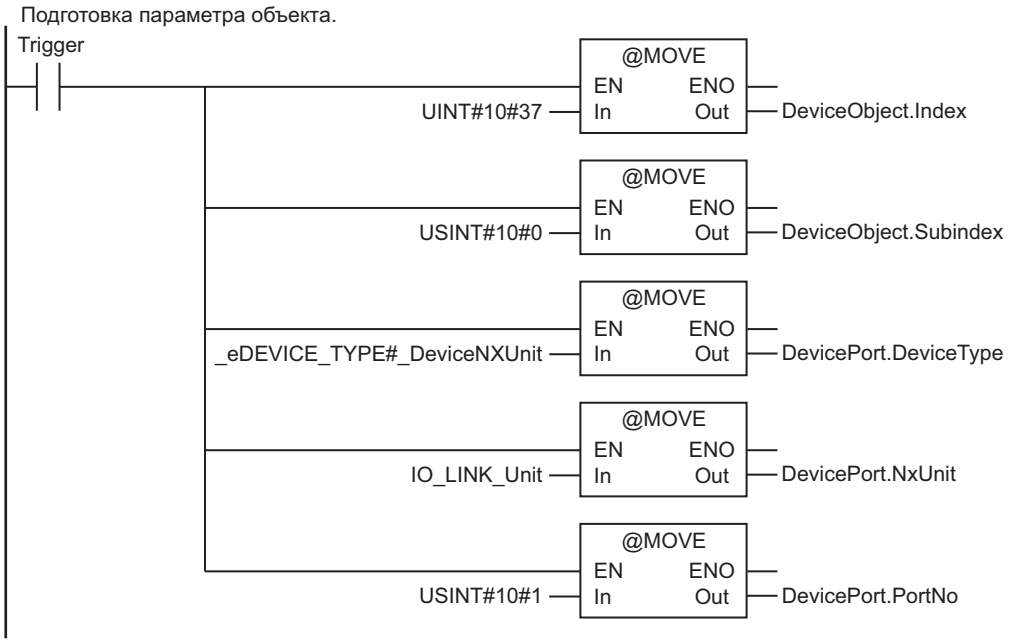
Из фотоэлектрического датчика (E3Z), подключенного к порту 1 модуля NX-ILM400, считывается журнал ошибок (индекс: 37 / подындекс: 0) размером 30 байт. Прочитанные данные сохраняются в *DeviceErrorLog*.
Адрес узла модуля NX-ECC203: 10.

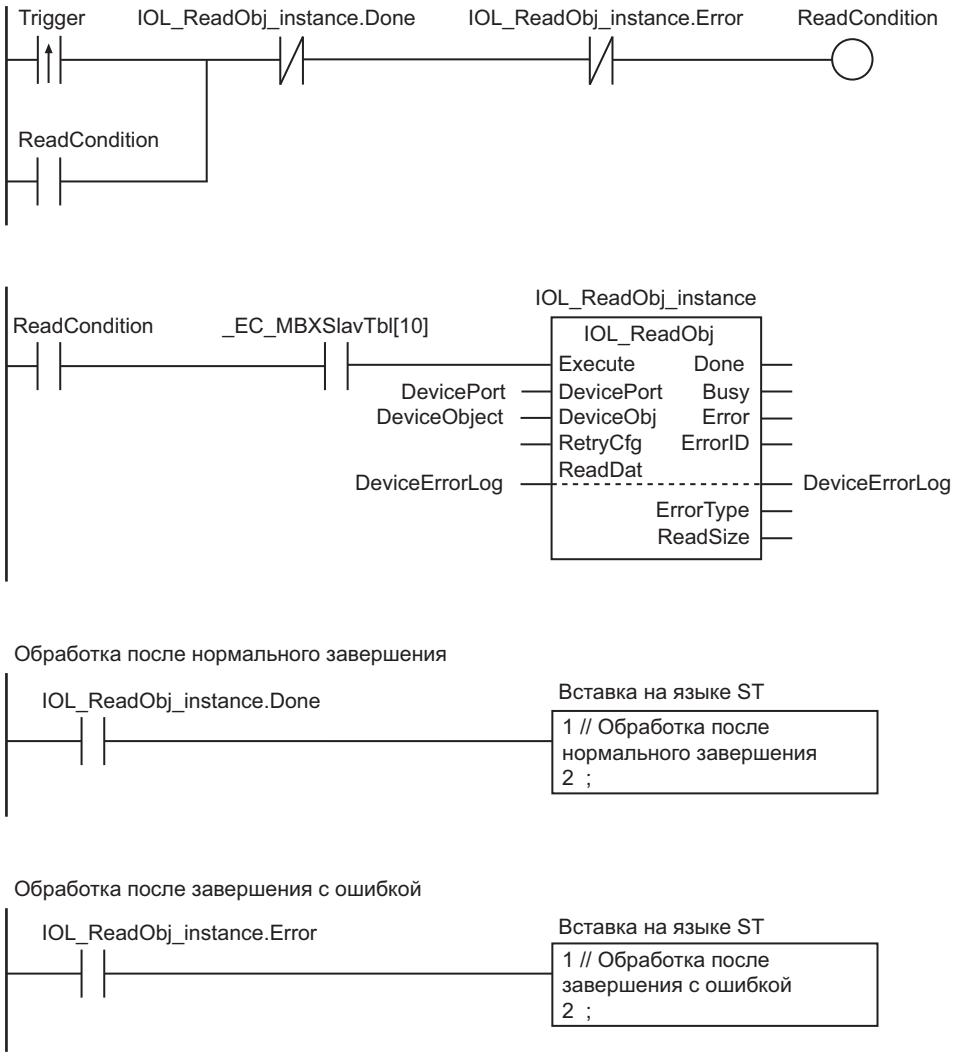
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	ReadCondition	BOOL	ЛОЖЬ	Условие выполнения чтения данных
	DevicePort	_sDEVICE_PORT		
	DeviceObject	_sIOLOBJ_ACCESS	(Index:=0, Subindex:=0)	Указание объекта устройства IO-Link
	DeviceErrorLog	ARRAY[1..30] OF BYTE		Прочитанные данные
	IOL_ReadObj_instance	IOL_ReadObj		

Внешние переменные	Переменная	Константа	Начальное значение	Комментарий
	_EC_MBXSlavTbl	☑	ARRAY[1..512] OF BOOL*1	Таблица ведомых устройств, способных участвовать в обмене сообщениями
	IO_LINK_Unit	☑	В качестве начального значения члена структуры *NxUnit* следует задать переменную устройства, которая указывает модуль NX-ILM400.	

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.





Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	ReadGo	BOOL	ЛОЖЬ	Выполнение чтения данных
	DevicePort	_sDEVICE_PORT		
	DeviceObject	_sIOLOBJ_ACCESS	(Index:=0, Subindex:=0)	Указание объекта устройства IO-Link
	DeviceErrorLog	ARRAY[1..30] OF BYTE		Прочитанные данные
	NormalEnd	UINT	0	Нормальное завершение
	ErrorEnd	UINT	0	Завершение с ошибкой
	R_Trig_Instance	R_Trig		
	IOL_ReadObj_instance	IOL_ReadObj		

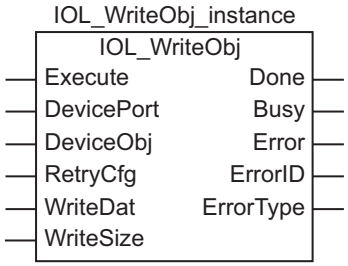
Внешние переменные	Переменная	Константа	Начальное значение	Комментарий
	_EC_MBXSlavTbl	☑	ARRAY[1..512] OF BOOL*1	Таблица ведомых устройств, способных участвовать в обмене сообщениями
	IO_LINK_Unit	☑	В качестве начального значения члена структуры *NxUnit* следует задать переменную устройства, которая указывает модуль NX-ILM400.	

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

```
// Подготовка параметра объекта.
R_Trig_instance(Clk := Trigger);
IF (R_Trig_instance.Q=TRUE)THEN
    DeviceObject.Index := UINT#10#37;
    DeviceObject.Subindex := USINT#0;
    DevicePort.DeviceType:= _eDEVICE_TYPE#_DeviceNXUnit;
    DevicePort.NxUnit:= IO_LINK_Unit;
    DevicePort.PortNo:= USINT#10#1;
    IF ( _EC_MBXSlavTbl[10] =TRUE)THEN
        ReadGo := TRUE;
    END_IF;
END_IF;

IF ( (IOL_ReadObj_instance.Done=TRUE) OR (IOL_ReadObj_instance.Error=TRUE) ) THEN
    ReadGo := FALSE;
END_IF;

// Выполнение команды IOL_ReadObj.
IOL_ReadObj_instance(
    Execute := ReadGo,
    DevicePort:= DevicePort,
    DeviceObj := DeviceObject,
    ReadDat :=DeviceErrorLog);

// Обработка после выполнения команды
IF (IOL_ReadObj_instance.Done=TRUE) THEN
    // Обработка после нормального завершения
    NormalEnd := NormalEnd + UINT#1;
ELSIF (IOL_ReadObj_instance.Error=TRUE) THEN
    // Обработка после завершения с ошибкой
    ErrorEnd := ErrorEnd + UINT#1;
END_IF;
```

IOL_WriteObj

Команда IOL_WriteObj производит запись данных в объекты устройств IO-Link.

Команда	Имя	FB /FUN	Графическое представление	Выражение языка ST
IOL_WriteObj	Запись в объект устройства IO-Link	FB		IOL_WriteObj_instance(Execute, DevicePort, DeviceObj, RetryCfg, WriteDat, WriteSize, Done, Busy, Error, ErrorID, ErrorType);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.12 или более поздней и Sysmac Studio версии 1.16 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
DeviceObj	Параметр объекта устройства IO-Link		Указание объекта устройства IO-Link	---	---	---
RetryCfg	Настройка повтора выполнения		Настройка повторных попыток выполнения команды	---	---	---
WriteDat	Записываемые данные		Данные, записываемые в устройство IO-Link	Зависит от типа данных.	---	---
WriteSize	Объем записываемых данных		Объем записываемых данных*1	10#1...10#232	Байты	---
ErrorType	Тип ошибки	Выход	Код ошибки, возвращаемый устройством IO-Link, сохраняется, когда *ErrorID = 4800 hex*.	16#0000...16#FFFF	---	---

*1. При записи значения логического типа (BOOL) введите *1*. При записи массива логического типа (BOOL) введите количество элементов.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1152.																			
DeviceObj	Подробные сведения о структуре _sIOLOBJ_ACCESS см. в разделе *Функция* на стр. 2-1152.																			
RetryCfg	Подробные сведения о структуре _sIOL_RETRY_CFG см. в разделе *Функция* на стр. 2-1152.																			
WriteDat	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также можно указать массив.																			
WriteSize							ОК													
ErrorType			ОК																	

Функция

Команда IOL_WriteObj производит запись данных объектов в устройства IO-Link.

Во входной переменной *DevicePort* следует указать модуль ведущего устройства IO-Link и номер порта, к которому подключено целевое устройство IO-Link, в которое записываются данные. Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Резерв	Резерв	Резерв	---	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2 3: порт 3 4: порт 4 5: порт 5 6: порт 6 7: порт 7 8: порт 8	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства. Укажите значение _DeviceNXUnit для модуля ведущего устройства IO-Link типа NX или значение _DeviceEcatSlave для модуля ведущего устройства IO-Link типа GX.

Переменная, используемая для указания устройства, определяется указанным типом устройства.

Для данной команды она определяется следующим образом:
Для указания устройства типа NX используется переменная *NxUnit*. Переменная *EcatSlave* в данном случае не используется.
В *NxUnit* следует передать переменную устройства, которая назначена указываемому устройству.
Для указания устройства типа GX используется переменная *EcatSlave*. Переменная *NxUnit* в данном случае не используется.
В *EcatSlave* следует передать переменную устройства, которая назначена указываемому устройству.

Используйте переменную *PortNo* для указания номера порта, к которому подключено устройство IO-Link.
Количество портов зависит от типа модуля ведущего устройства IO-Link.
Модуль типа NX: 1…4
Модуль типа GX: 1…8

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.

Используйте входную переменную *DeviceObj* для указания параметра объекта для устройства IO-Link, в которое записываются данные.

Для входной переменной *DeviceObj* используется структурный тип данных _sIOLOBJ_ACCESS.
Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DeviceObj	Параметр объекта устройства IO-Link	Указание объекта устройства IO-Link	_sIOLOBJ_ACCESS	---	---	---
Index	Индекс	Индекс	UINT	Зависит от типа данных.	---	---
Subindex	Подындекс	Задайте значение *0* для чтения из всего индекса.	USINT	Зависит от типа данных.	---	---

Используйте входную переменную *RetryCfg* для настройки повторных попыток выполнения команды.
Для переменной *RetryCfg* используется структурный тип данных _sIOL_RETRY_CFG. Описание приведено в таблице ниже.

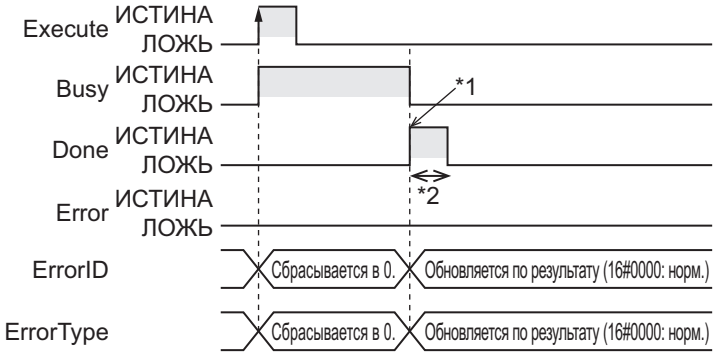
Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
RetryCfg	Настройка повтора выполнения	Настройка повторных попыток выполнения команды	_sIOL_RETRY_CFG	---	---	---
TimeOut	Время ожидания	Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	TIME	0...300 с	---	T#2.0s
RetryNum	Количество повторных попыток	Количество повторных попыток при превышении времени ожидания Если задано значение *0*, предпринимается 3 повторных попытки.	UINT	Зависит от типа данных.	Кол-во раз	3

Используйте входную переменную *WriteDat* для указания данных, которые должны быть записаны в устройство IO-Link.

Временные диаграммы

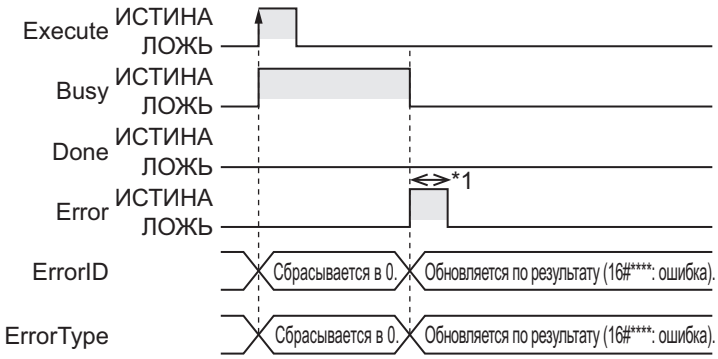
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Запись завершена.
*2. Период выполнения задачи

● Завершение с ошибкой



*1. Период выполнения задачи

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EC_MBXSlavTbl	Таблица ведомых устройств, способных участвовать в обмене сообщениями	ARRAY[1..512] OF BOOL*1	В этой таблице указываются ведомые устройства, которые способны участвовать в обмене сообщениями. Таблица содержит данные для ведомых устройств в порядке возрастания адресов узлов. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: ARRAY [1..192] OF BOOL.

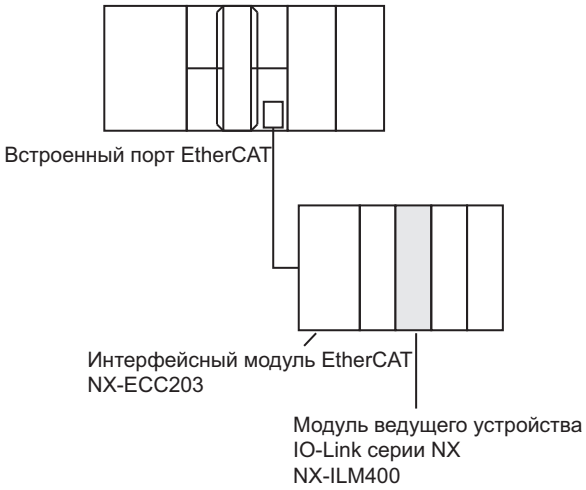
Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Для параметров *DevicePort.NxUnit* и *DevicePort.EcatSlave* должна быть указана соответствующая переменная устройства, которая назначена модулю ведущего устройства IO-Link на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio. Информацию о назначении переменных устройства см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.
- Всегда используйте переменную в качестве входного параметра, передаваемого в *WriteDat*. При передаче константы произойдет ошибка сборки.
- Для модуля ведущего устройства IO-Link, независимо от его типа (NX или GX), можно выполнять только одну команду за раз.
- Эту команду невозможно использовать в событийной задаче. В этом случае произойдет ошибка компиляции.
- Эта команда выполняется, когда вход *Execute* переходит в состояние ИСТИНА. Команда не выполняется, если вход *Execute* постоянно находится в состоянии ИСТИНА.
- Для команд IOL_ReadObj и IOL_WriteObj можно определить максимум 64 экземпляра.
- В указанных ниже случаях произойдет ошибка.
  a) Для параметра *DevicePort.NxUnit* или *DevicePort.EcatSlave* указано значение, выходящее за пределы допустимого диапазона.
  b) Значение *TimeOut* находится за пределами допустимого диапазона.
  c) Недопустимый тип данных переменной *DevicePort*.
  d) Для *WriteDat* указано более 232 байтов данных.
  e) От устройства IO-Link получен ответ с кодом ошибки.
  Старшие восемь битов представляют код ошибки (*ErrorCode*), а младшие восемь битов представляют дополнительный код (*AdditionalCode*).
  Сведения о значениях *ErrorCode* и *AdditionalCode* см. в описании типов ошибок в документе «IO-Link Communication Specification» («Техническое описание интерфейса связи IO-Link»). Описание типов ошибок можно найти на веб-сайте консорциума IO-Link: http://www.io-link.com/
  f) Указанного модуля ведущего устройства IO-Link не существует.

g) Превышено максимальное количество сообщений, которые может обработать ведущее устройство IO-Link. Команда не может быть выполнена, так как ведущее устройство IO-Link обрабатывает сообщения от других приложений.
h) Указанный модуль ведущего устройства IO-Link не находится в состоянии, в котором он может принимать сообщения.
i) Было выполнено более 32 следующих команд одновременно: EC_CoESDOWrite, EC_CoESDORead, EC_StartMon, EC_StopMon, EC_SaveMon, EC_CopyMon, EC_DisconnectSlave, EC_ConnectSlave, EC_ChangeEnableSetting, IOL_ReadObj и IOL_WriteObj.
j) Превышено время ожидания при обмене данными.
k) Указанный порт модуля ведущего устройства IO-Link не находится в режиме IO-Link. Порт отключен или находится в режиме SIO.
l) Устройство IO-Link не подключено к указанному порту модуля ведущего устройства IO-Link.
m) На указанный порт модуля ведущего устройства IO-Link не подается питание ввода-вывода.
n) В указанном порте модуля ведущего устройства IO-Link произошла ошибка проверки или ошибка связи.

Пример программы

В данном примере модуль ведущего устройства IO-Link (NX-ILM400) подключен к интерфейсному модулю EtherCAT (NX-ECC203).



Записывается значение *01* в однобайтовый логический выход 1 SwitchPoint (индекс: 61/подындекс: 1) фотоэлектрического датчика (E3Z), подключенного к порту 1 модуля NX-ILM400. Записываемые данные хранятся в переменной *SwitchPoint*.
Адрес узла модуля NX-ECC203: 10.

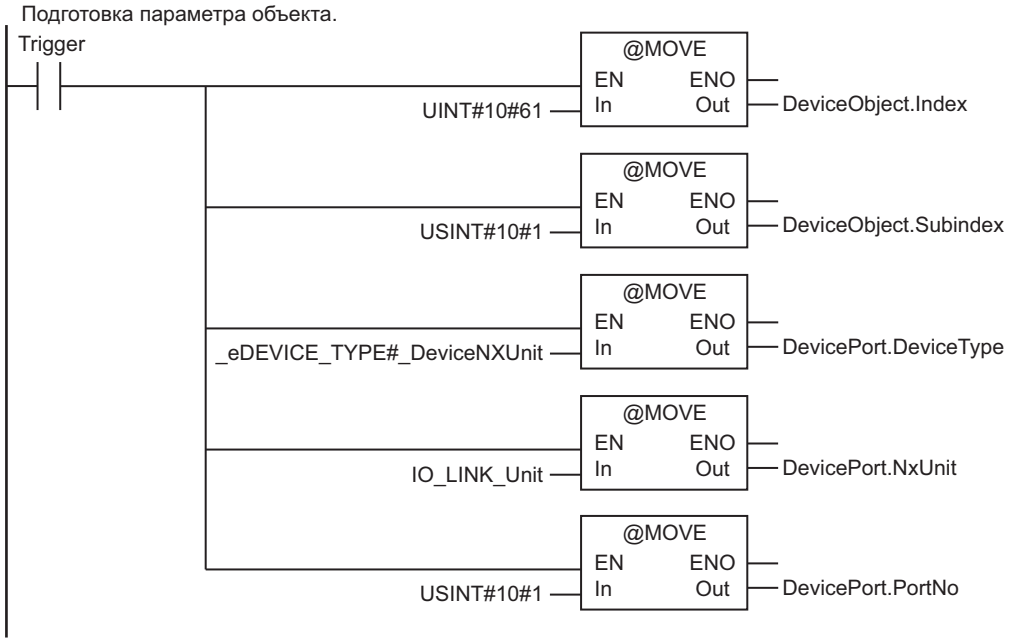
Программа на языке LD

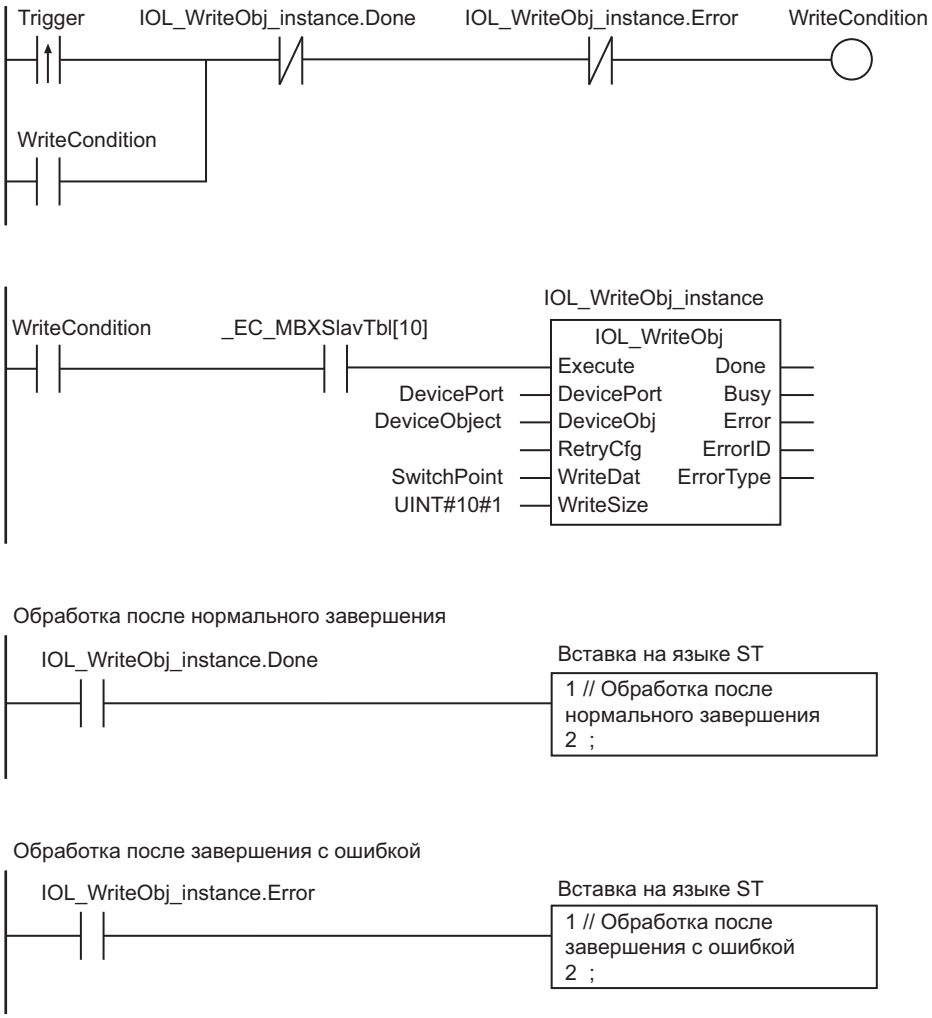
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	WriteCondition	BOOL	ЛОЖЬ	Условие выполнения записи данных
	DevicePort	_sDEVICE_PORT		
	DeviceObject	_sIOLOBJ_ACCESS	(Index:=0, Subindex:=0)	Указание объекта устройства IO-Link
	SwitchPoint	USINT	USINT#01	Записываемые данные
	IOL_WriteObj_instance	IOL_WriteObj		

Внешние переменные	Переменная	Константа	Начальное значение	Комментарий
	_EC_MBXSlavTbl	☑	ARRAY[1..512] OF BOOL*1	Таблица ведомых устройств, способных участвовать в обмене сообщениями
	IO_LINK_Unit	☑	В качестве начального значения члена структуры *NxUnit* следует задать переменную устройства, которая указывает модуль NX-ILM400.	

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.





Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	WriteGo	BOOL	ЛОЖЬ	Выполнение записи данных
	DevicePort	_sDEVICE_PORT		
	DeviceObject	_sIOLOBJ_ACCESS	(Index:=0, Subindex:=0)	Указание объекта устройства IO-Link
	SwitchPoint	USINT	USINT#01	Записываемые данные
	NormalEnd	UINT	0	Нормальное завершение
	ErrorEnd	UINT	0	Завершение с ошибкой
	R_Trig_Instance	R_Trig		
	IOL_WriteObj_instance	IOL_WriteObj		

Внешние переменные	Переменная	Константа	Начальное значение	Комментарий
	_EC_MBXSlavTbl	☑	ARRAY[1..512] OF BOOL*1	Таблица ведомых устройств, способных участвовать в обмене сообщениями
	IO_LINK_Unit	☑	В качестве начального значения члена структуры *NxUnit* следует задать переменную устройства, которая указывает модуль NX-ILM400.	

*1. Тип данных для модулей ЦПУ NX102 и NX1P2, а также для модулей ЦПУ серии NJ: *ARRAY[1..192] OF BOOL*.

```
// Подготовка параметра объекта.
R_Trig_instance(Clk := Trigger);
IF (R_Trig_instance.Q=TRUE)THEN
    DeviceObject.Index := UINT#10#61;
    DeviceObject.Subindex := USINT#1;
    DevicePort.DeviceType:= _eDEVICE_TYPE#_DeviceNXUnit;
    DevicePort.NxUnit:= IO_LINK_Unit;
    DevicePort.PortNo:= USINT#10#1;
    IF ( _EC_MBXSlavTbl[10] =TRUE)THEN
        WriteGo := TRUE;
    END_IF;
END_IF;

IF ( (IOL_WriteObj_instance.Done=TRUE) OR (IOL_WriteObj_instance.Error=TRUE) ) THEN
    WriteGo := FALSE;
END_IF;

// Выполнение команды IOL_WriteObj.
IOL_WriteObj_instance(
    Execute := WriteGo,
    DevicePort:= DevicePort,
    DeviceObj := DeviceObject,
    WriteDat := SwitchPoint,
    WriteSize := UINT#10#1);

// Обработка после выполнения команды
IF (IOL_WriteObj_instance.Done=TRUE) THEN
    // Обработка после нормального завершения
    NormalEnd := NormalEnd + UINT#1;
ELSIF (IOL_WriteObj_instance.Error=TRUE) THEN
    // Обработка после завершения с ошибкой
    ErrorEnd := ErrorEnd + UINT#1;
END_IF;
```

Команды для обмена данными по интерфейсу EtherNet/IP

Команда	Имя	Стр.
CIPOpen	Открытие CIP-соединения класса 3 (Large_Forward_Open)	стр. 2-1163
CIPOpenWithDataSize	Открытие CIP-соединения класса 3 с указанным объемом данных	стр. 2-1174
CIPRead	Чтение переменной посредством явного сообщения класса 3	стр. 2-1178
CIPWrite	Запись переменной посредством явного сообщения класса 3	стр. 2-1184
CIPSend	Передача явного сообщения класса 3	стр. 2-1191
CIPClose	Закрытие CIP-соединения класса 3	стр. 2-1197
CIPUCMMRead	Чтение переменной посредством явного сообщения UCMM	стр. 2-1200
CIPUCMMWrite	Запись переменной посредством явного сообщения UCMM	стр. 2-1206
CIPUCMMSend	Передача явного сообщения UCMM	стр. 2-1214
SktUDPCreate	Создание сокета UDP	стр. 2-1227
SktUDPRcv	Прием через сокет UDP	стр. 2-1236
SktUDPSend	Передача через сокет UDP	стр. 2-1240
SktTCPAccept	Принятие сокета TCP	стр. 2-1243
SktTCPConnect	Соединение с сокетом TCP	стр. 2-1246
SktTCPRcv	Прием через сокет TCP	стр. 2-1255
SktTCPSend	Передача через сокет TCP	стр. 2-1259
SktGetTCPStatus	Чтение состояния сокета TCP	стр. 2-1262
SktClose	Закрытие сокета TCP/UDP	стр. 2-1265
SktClearBuf	Очистка буфера приема сокета TCP/UDP	стр. 2-1268
SktSetOption	Настройка дополнительного параметра сокета TCP	стр. 2-1271
ModbusTCPCmd	Передача команды общего назначения по протоколу Modbus TCP	стр. 2-1277
ModbusTCPRead	Передача команды чтения по протоколу Modbus TCP	стр. 2-1286
ModbusTCPWrite	Передача команды записи по протоколу Modbus TCP	стр. 2-1295

Команда	Имя	Стр.
ChangeIPAdr	Изменение IP-адреса	стр. 2-1303
ChangeFTPAccount	Изменение учетной записи FTP	стр. 2-1312
ChangeNTPServerAdr	Изменение адреса сервера NTP	стр. 2-1316
FTPGetFileList	Получение списка файлов на сервере FTP	стр. 2-1321
FTPGetFile	Получить файл с сервера FTP	стр. 2-1337
FTPPutFile	Поместить файл на сервер FTP	стр. 2-1347
FTPRemoveFile	Удаление файла с сервера FTP	стр. 2-1358
FTPRemoveDir	Удаление каталога с сервера FTP	стр. 2-1369

CIPOpen

Команда CIPOpen открывает соединение класса 3 (Large_Forward_Open) по протоколу CIP с указанным удаленным узлом. Длина данных устанавливается равной 1994 байт.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CIPOpen	Открытие CIP-соединения класса 3 (Large_Forward _Open)	FB	CIPOpen_instance CIPOpen Execute — Done RoutePath — Busy TimeOut — Error ErrorID ErrorIDEx Handle	CIPOpen_instance(Execute, RoutePath, TimeOut, Done, Busy, Error, ErrorID, ErrorIDEx, Handle);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
RoutePath	Путь маршрута	Вход	Путь маршрута	Зависит от типа данных.	---	---
TimeOut	Время ожидания		Время ожидания	1…65535	0,1 с	20 (2 с)
Handle	Дескриптор	Выход	Дескриптор	---	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RoutePath																				ОК
TimeOut							ОК													
Handle	Подробные сведения о структуре _sCIP_HANDLE см. в разделе *Функция* на стр. 2-1163.																			

Функция

Команда CIPOpen открывает соединение класса 3 (Large_Forward_Open) по протоколу CIP с удаленным узлом в сети CIP. Удаленный узел указывается с помощью параметра *RoutePath* (путь маршрута). Длина данных устанавливается равной 1994 байт.
Когда соединение открыто, в переменную *Handle* выводится значение дескриптора.

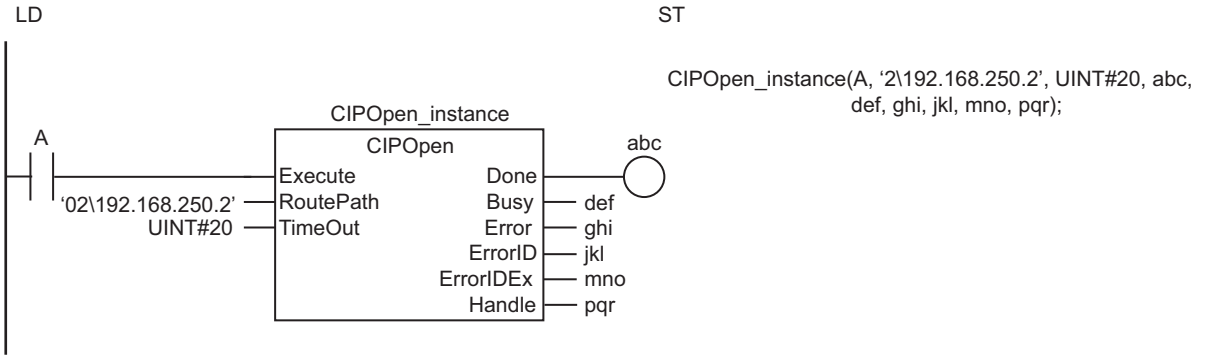
В параметре *TimeOut* задается время ожидания соединения.
Если после выполнения команды CIPSend, CIPWrite или CIPRead ответ от удаленного узла не поступает в течение заданного времени ожидания соединения, считается, что произошел сбой связи.

Если при выполнении команды CIPRead, CIPWrite или CIPSend удаленный узел возвращает ответ, время ожидания соединения сбрасывается.

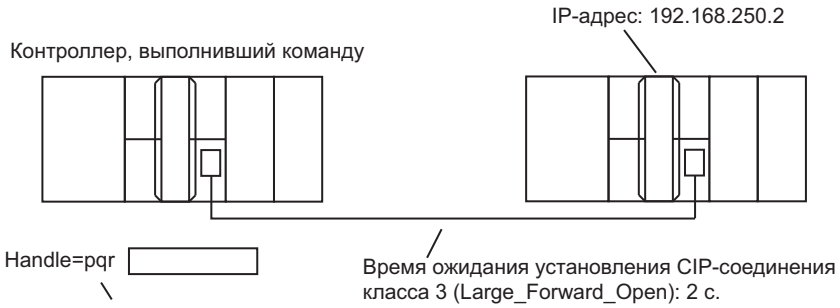
Для переменной *Handle* используется структурный тип данных _sCIP_HANDLE. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Handle	Дескриптор	Дескриптор	_sCIP_HANDLE	---	---	---
Handle	Дескриптор	Дескриптор	UDINT	Зависит от типа данных.	---	---

Ниже приведен пример, в котором *RoutePath* = «02\192.168.250.2», а *TimeOut* = UINT#20. Команда CIPOpen (Открытие CIP-соединения класса 3 (Large_Forward_Open)) открывает соединение класса 3 по протоколу CIP с удаленным узлом с IP-адресом 192.168.250.2. Время ожидания составляет 2 с. Значение дескриптора присваивается переменной *pqr*.



Команда CIPOpen (Открытие CIP-соединения класса 3 (Large_Forward_Open)) открывает соединение класса 3 по протоколу CIP с удаленным узлом в сети CIP. Удаленный узел указывается с помощью параметра ***RoutePath***.



Если значение *ErrorID* = WORD#16#1C00, в переменную *ErrorIDEx* сохраняется код ошибки сообщения CIP. Значение кода ошибки в *ErrorIDEx* и его смысл зависят от конкретного удаленного узла. См. руководство по соответствующему удаленному узлу.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

- Дополнительные сведения об интерфейсе связи CIP см. в следующих документах.
  a) *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*
  b) *Серия CJ, модули EtherNet/IP — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W495)*
- Для установления соединения Forward Open или соединения с любой заданной длиной данных используйте команду *CIPOpenWithDataSize* на стр. 2-1174.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эта команда или команда CIPOpenWithDataSize должна выполняться перед выполнением команды CIPRead, CIPWrite или CIPSend.
- В случае этой команды первое значение времени ожидания после установления соединения принимается равным 10 с, даже если для параметра *TimeOut* задано значение меньше 100 (10 с).
- Для закрытия соединений, открытых с помощью команды CIPOpen, используйте команду CIPClose.
- Созданный этой командой дескриптор сохраняется, даже если время ожидания соединения оказывается превышено. Обязательно используйте команду CIPClose для закрытия соединения.
- При переходе в режим «Программирование» созданные этой командой дескрипторы деактивируются.
- Одновременно может быть создано не более 32 дескрипторов.
- Эту команду можно использовать для встроенного порта EtherNet/IP модуля ЦПУ серии NJ/NX или порта модуля интерфейса EtherNet/IP, подключенного к модулю ЦПУ серии NJ.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение *TimeOut* находится за пределами допустимого диапазона.
  b) В параметр *RoutePath* передана недопустимая текстовая строка.
  c) Одновременно было выполнено более 32 команд, связанных с протоколом CIP.

d) Была предпринята попытка открыть соединение сверх допустимого количества соединений CIPClass (32 соединения).
e) Не был получен ответ на запрос открытия соединения.
f) Удаленный узел, с которым открывается соединение, не поддерживает соединения Large_Forward_Open.
g) Имеется ошибка в настройке локального IP-адреса.
h) Произошла ошибка дублирования IP-адреса.
i) Все TCP-соединения уже используются.
j) Команда была выполнена при ошибке сервера BOOTP.

Информация о версии

При использовании модуля ЦПУ версии 1.10 или более поздней версии значение *Handle* не изменяется, даже если значение *Error* меняется на ИСТИНА. В случае версии 1.09 или более ранней версии значение *Handle* меняется на *0*.

Пример программы

В данном примере используются сообщения класса 3 протокола CIP для записи значения в переменную, чтения значения переменной и передачи сообщения.
Контроллеры подключены к сети EtherNet/IP. IP-адрес удаленного узла: 192.168.250.2.
Соблюдается приведенный ниже порядок действий.

1 Используется команда CIPOpen для открытия соединения класса 3 (Large_Forward_Open). Время ожидания составляет 2 с.

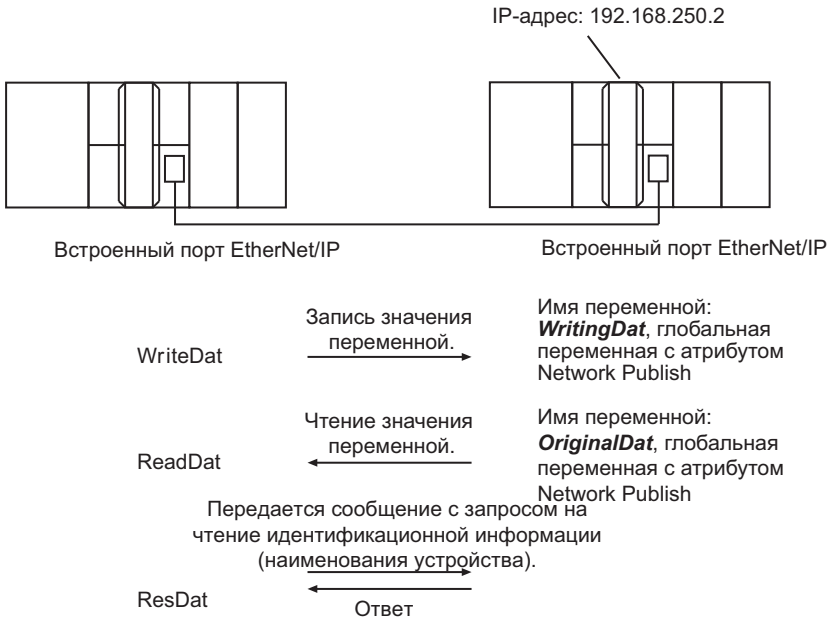
2 Используется команда CIPWrite для записи значения переменной на удаленном узле. Переменная на удаленном узле носит имя *WritingDat*, в нее записывается содержимое переменной *WriteDat*.
Переменная *WritingDat* должна быть определена на удаленном узле как глобальная переменная, и для нее должен быть настроен атрибут Network Publish (Публикация в сети).

3 Используется команда CIPRead для чтения значения переменной на удаленном узле. Производится чтение значения переменной *OriginalDat* на другом узле, и прочитанное значение сохраняется в переменную *ReadDat*.
Переменная *OriginalDat* должна быть определена на удаленном узле как глобальная переменная, и для нее должен быть настроен атрибут Network Publish (Публикация в сети).

4 Используется команда CIPSend для передачи явного сообщения удаленному узлу. Сообщение содержит запрос на чтение идентификационной информации (наименования устройства).
Значения идентификатора класса, идентификатора экземпляра, идентификатора атрибута и кода службы приведены в таблице ниже. Возвращенные данные сохраняются в переменную *ResDat*.

Параметр	Значение
Идентификатор класса	1
Идентификатор экземпляра	1
Идентификатор атрибута	7
Код службы	16#0E

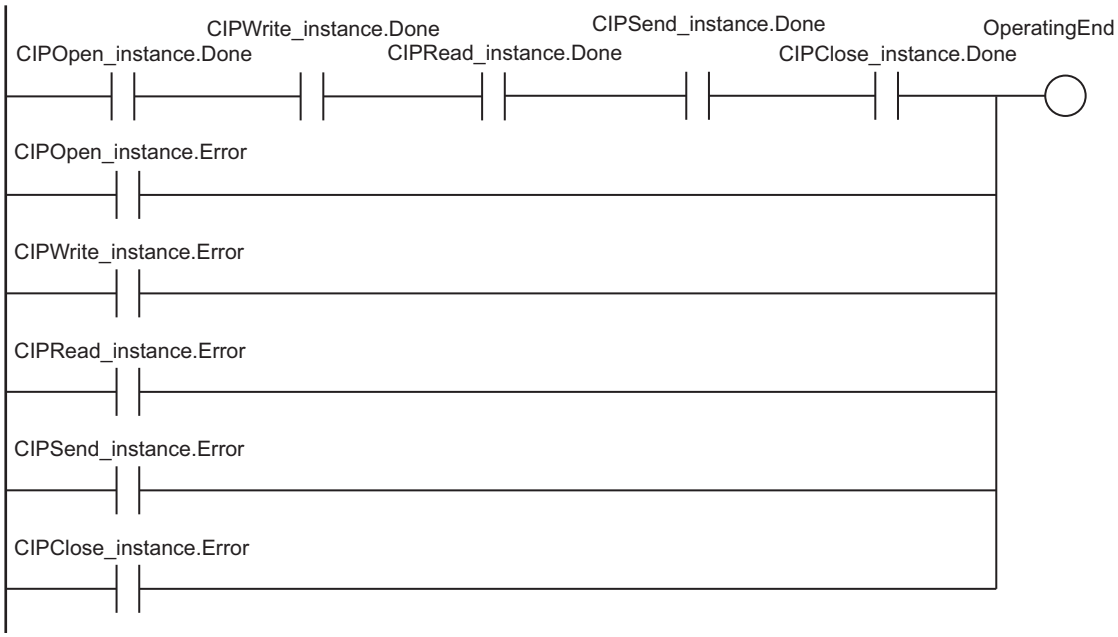
5 Используется команда CIPClose для закрытия соединения класса 3.



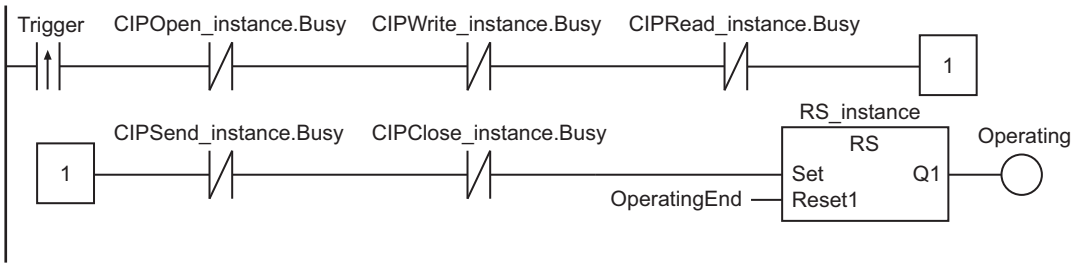
Программа на языке LD

Переменная	Тип данных	Начальное значение	Комментарий
OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена
Trigger	BOOL	ЛОЖЬ	Условие выполнения
Operating	BOOL	ЛОЖЬ	Обработка
WriteDat	INT	1234	Записываемые данные
ReadDat	INT	0	Прочитанные данные
ReqPath	_sREQUEST_PATH	(ClassID:=0, InstanceID:=0, isAttributeID:=FALSE, AttributeID:=0)	Путь запроса
ResDat	ARRAY[0..10] OF BYTE	[11(16#0)]	Данные ответа
Dummy	BYTE	16#0	Формальная переменная
RS_instance	RS		
CIPOpen_instance	CIPOpen		
CIPWrite_instance	CIPWrite		
CIPRead_instance	CIPRead		
CIPSend_instance	CIPSend		
CIPClose_instance	CIPClose		

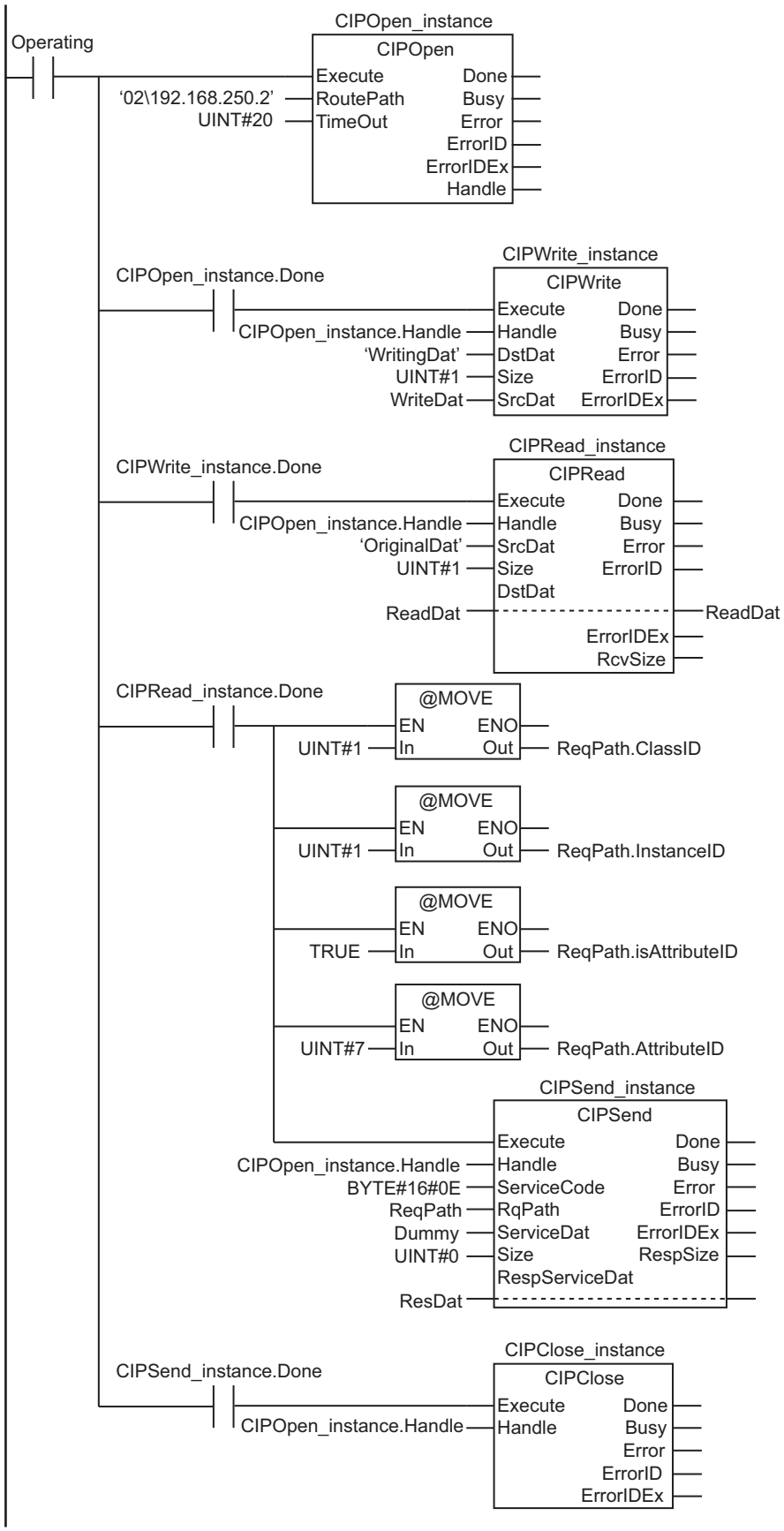
Определение, завершено ли выполнение команды.

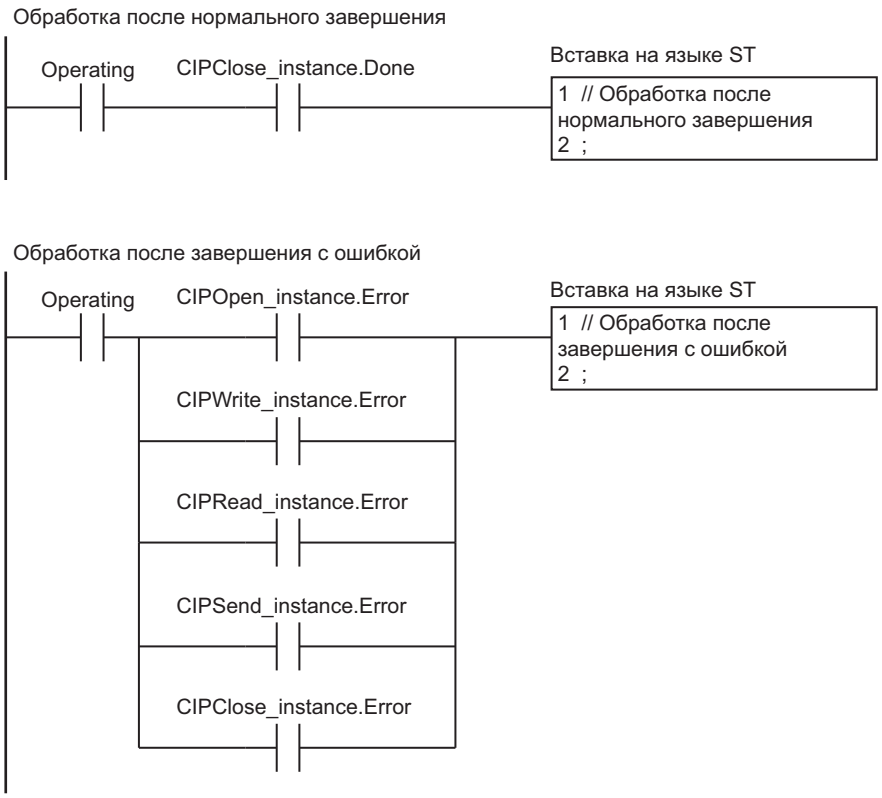


Прием условия выполнения.



Выполнение команды





Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	DoCIPTrigger	BOOL	ЛОЖЬ	Обработка
	Stage	INT	0	Смена этапа
	WriteDat	INT	0	Записываемые данные
	ReadDat	INT	0	Прочитанные данные
	ReqPath	_sREQUEST_PATH	(ClassID:=0, InstanceID:=0, isAttributeID:=FALSE, AttributeID:=0)	Путь запроса
	ResDat	ARRAY[0..10] OF BYTE	[11(16#0)]	Данные ответа
	Dummy	BYTE	16#0	Формальная переменная
	CIPOpen_instance	CIPOpen		
	CIPWrite_instance	CIPWrite		
	CIPRead_instance	CIPRead		
	CIPSend_instance	CIPSend		
	CIPClose_instance	CIPClose		

Внешние переменные	Переменная	Константа	Тип данных	Комментарий
	_EIP_EtnOnlineSta	☑	BOOL	В сети

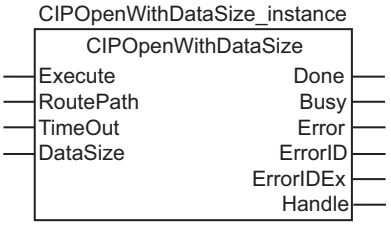
```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (DoCIPTrigger=FALSE) AND (_Eip_EtnOnlineSta=TRUE) ) THEN
    DoCIPTrigger  :=TRUE;
    Stage  :=INT#1;
    CIPOpen_instance(Execute:=FALSE);  // Инициализация экземпляра.
    CIPWrite_instance(
        Execute  :=FALSE,  // Инициализация экземпляра.
        SrcDat  :=WriteDat);  // Формальная запись
    CIPRead_instance(  // Инициализация экземпляра.
        Execute  :=FALSE,  // Формальная запись
        DstDat  :=ReadDat);  // Формальная запись
    CIPSend_instance(
        Execute  :=FALSE,  // Инициализация экземпляра.
        ServiceDat  := Dummy,  // Формальная запись
        RespServiceDat  :=ResDat);  // Формальная запись
    CIPClose_instance(Execute:=FALSE);  // Инициализация экземпляра.
END_IF;

IF (DoCIPTrigger=TRUE) THEN
    CASE Stage OF
    1 :  // Открытие CIP-соединения класса 3 (Large_Forward_Open)
        CIPOpen_instance(
            Execute  :=TRUE,
            TimeOut  :=UINT#20,  // Время ожидания: 2,0 с
            RoutePath  :='02\192.168.250.2');  // Путь маршрута

        IF (CIPOpen_instance.Done=TRUE) THEN
            Stage:=INT#2;  // Нормальное завершение
        ELSIF (CIPOpen_instance.Error=TRUE) THEN
            Stage:=INT#10;  // Завершение с ошибкой
        END_IF;

    2 :  // Запрос записи значения переменной.
        CIPWrite_instance(
            Execute  :=TRUE,
            Handle  :=CIPOpen_instance.Handle,  // Дескриптор
            DstDat  :='WritingDat',  // Имя адресуемой переменной
            Size    :=UINT#1,  // Количество записываемых элементов
            SrcDat  :=WriteDat);  // Записываемые данные
        IF (CIPWrite_instance.Done=TRUE) THEN
            Stage:=INT#3;  // Нормальное завершение
        ELSIF (CIPWrite_instance.Error=TRUE) THEN
            Stage:=INT#20;  // Завершение с ошибкой
        END_IF;
```

```
3 :  // Запрос чтения значения переменной.
        CIPRead_instance(
            Execute  :=TRUE,
            Handle  :=CIPOpen_instance.Handle,  // Дескриптор
            SrcDat  :='OriginalDat',  // Имя адресуемой переменной
            Size  :=UINT#1,  // Количество элементов для чтения
            DstDat  :=ReadDat);  // Прочитанные данные

        IF (CIPRead_instance.Done=TRUE) THEN
            Stage:=INT#4;  // Нормальное завершение
        ELSIF (CIPRead_instance.Error=TRUE) THEN
            Stage:=INT#30;  // Завершение с ошибкой
        END_IF;

4 :  // Передача сообщения
    ReqPath.ClassID  :=UINT#01;
    ReqPath.InstanceID  :=UINT#01;
    ReqPath.isAttributeID  :=TRUE;
    ReqPath.AttributeID  :=UINT#07;
    CIPSend_instance(
        Execute  :=TRUE,
        Handle  :=CIPOpen_instance.Handle,  // Дескриптор
        ServiceCode  :=BYTE#16#0E,  // Код службы
        RqPath  :=ReqPath,  // Путь запроса
        ServiceDat  :=Dummy,  // Данные службы
        Size  :=UINT#0,  // Количество элементов
        RespServiceDat:=ResDat);  // Данные ответа

    IF (CIPSend_instance.Done=TRUE) THEN
        Stage:=INT#5;  // Нормальное завершение
    ELSIF (CIPSend_instance.Error=TRUE) THEN
        Stage:=INT#40;  // Завершение с ошибкой
    END_IF;

5 :  // Запрос закрытия CIP-соединения класса 3.
    CIPClose_instance(
        Execute  :=TRUE,
        Handle  :=CIPOpen_instance.Handle);  // Дескриптор

    IF (CIPClose_instance.Done=TRUE) THEN
        Stage:=INT#0;
    ELSIF (CIPClose_instance.Error=TRUE) THEN
        Stage:=INT#50;
    END_IF;

0:  // Обработка после нормального завершения
    DoCIPTrigger  :=FALSE;
    Trigger  :=FALSE;

ELSE  // Обработка после завершения с ошибкой
```

```
        DoCIPTrigger  :=FALSE;
        Trigger  :=FALSE;
    END_CASE;
END_IF;
```

CIPOpenWithDataSize

Команда CIPOpenWithDataSize открывает соединение класса 3 по протоколу CIP с указанным удаленным узлом, которое позволяет передавать и принимать явные сообщения класса 3 с указанной длиной данных или меньшей длиной.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CIPOpenWith DataSize	Открытие CIP-соединения класса 3 с указанным объемом данных	FB		CIPOpen_instance(Execute, RoutePath, TimeOut, DataSize, Done, Busy, Error, ErrorID, ErrorIDEx, Handle);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.06 или более поздней и Sysmac Studio версии 1.07 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
RoutePath	Путь маршрута	Вход	Путь маршрута	Зависит от типа данных.	---	---
TimeOut	Время ожидания		Время ожидания	1…65 535	0,1 с	20 (2 с)
DataSize	Длина данных		Длина данных	6...8192*1 *2	Байты	1994
Handle	Дескриптор	Выход	Дескриптор	---	---	---

*1. Возможные значения для модулей ЦПУ NX1P2 и модулей ЦПУ серии NJ: *6...1 994*.

*2. В случае модуля ЦПУ с версией модуля 1.10 или более ранней или Sysmac Studio версии 1.14 или ниже минимальное значение равно *10*.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RoutePath																				OK
TimeOut							OK													
DataSize							OK													
Handle	Подробные сведения о структуре _sCIP_HANDLE см. в разделе *Функция* на стр. 2-1175.																			

Функция

Команда CIPOpenWithDataSize открывает соединение класса 3 по протоколу CIP с удаленным узлом в сети CIP. Удаленный узел указывается с помощью параметра *RoutePath* (путь маршрута). В параметре *DataSize* (длина данных) указывается объем данных, которые могут быть переданы или приняты с использованием явного сообщения класса 3.
От значения *DataSize* зависит используемая служба соединений класса 3, что показано в таблице ниже.

Значение *DataSize* [байт]	Служба
509 или меньше	Forward_Open
510...8 192*1	Large_Forward_Open

*1. Возможные значения для модулей ЦПУ NX1P2 и модулей ЦПУ серии NJ: *510...1 994*.

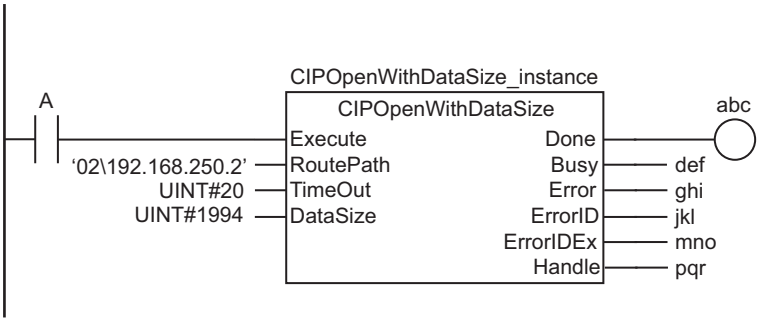
Когда соединение открыто, в переменную *Handle* выводится значение дескриптора.
В параметре *TimeOut* задается время ожидания соединения. Если после выполнения команды CIPSend, CIPWrite или CIPRead ответ от удаленного узла не поступает в течение заданного времени ожидания соединения, считается, что произошел сбой связи.
Если при выполнении команды CIPRead, CIPWrite или CIPSend удаленный узел возвращает ответ, время ожидания соединения сбрасывается.

Для переменной *Handle* используется структурный тип данных _sCIP_HANDLE. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Handle	Дескриптор	Дескриптор	_sCIP_HANDLE	---	---	---
Handle	Дескриптор	Дескриптор	UDINT	Зависит от типа данных.	---	---

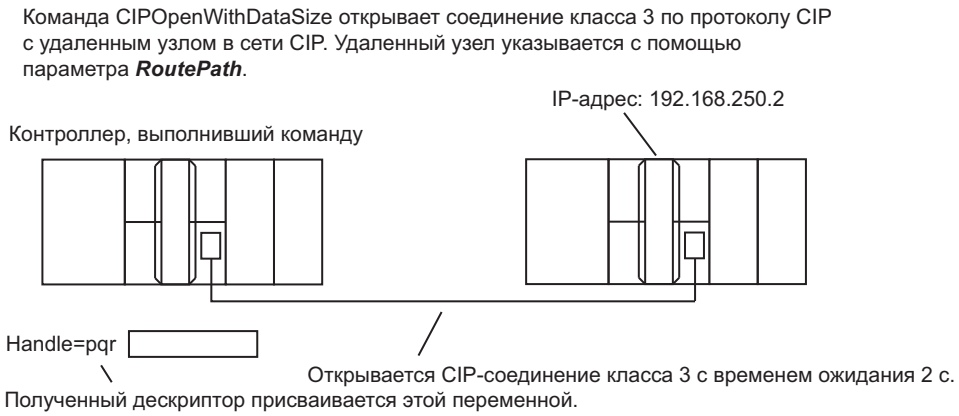
Ниже приведен пример, в котором *RoutePath* = «02\192.168.250.2», а *TimeOut* = UINT#20.
Команда CIPOpenWithDataSize открывает соединение класса 3 по протоколу CIP с удаленным узлом с IP-адресом 192.168.250.2. Длина данных равна 1994 байт, а время ожидания составляет 2 с. Значение дескриптора присваивается переменной *pqr*.

LD



ST

```
CIPOpenWithDataSize_instance(A,
                             '2\192.168.250.2',
                             UINT#20, UINT#1994,
                             abc, def, ghi, jkl, mno,
                             pqr);
```



Если значение *ErrorID* = WORD#16#1C00, в переменную *ErrorIDEx* сохраняется код ошибки сообщения CIP. Значение кода ошибки в *ErrorIDEx* и его смысл зависят от конкретного удаленного узла. См. руководство по соответствующему удаленному узлу.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

- Дополнительные сведения об интерфейсе связи CIP см. в следующих документах.
  a) *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*
  b) *Серия CJ, модули EtherNet/IP — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W495)*
- Также можно использовать команду *CIPOpen* на стр. 2-1163 для использования Large_Forward_Open в качестве службы соединения класса 3.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эта команда или команда CIPOpen должна выполняться перед выполнением команды CIPRead, CIPWrite или CIPSend.

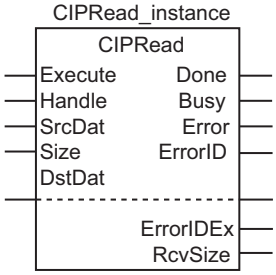
- В случае этой команды первое значение времени ожидания после установления соединения принимается равным 10 с, даже если для параметра *TimeOut* задано значение меньше 100 (10 с).
- Для закрытия соединений, открытых с помощью команды CIPOpenWithDataSize, используйте команду CIPClose.
- Созданный этой командой дескриптор сохраняется, даже если время ожидания соединения оказывается превышено. Обязательно используйте команду CIPClose для закрытия соединения.
- При переходе в режим «Программирование» созданные этой командой дескрипторы деактивируются.
- Одновременно может быть создано не более 32 дескрипторов.
- Эту команду можно использовать для встроенного порта EtherNet/IP модуля ЦПУ серии NJ/NX или порта модуля интерфейса EtherNet/IP, подключенного к модулю ЦПУ серии NJ.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение *TimeOut* находится за пределами допустимого диапазона.
  b) В параметр *RoutePath* передана недопустимая текстовая строка.
  c) Одновременно было выполнено более 32 команд, связанных с протоколом CIP.
  d) Была предпринята попытка открыть соединение сверх допустимого количества соединений CIPClass (32 соединения).
  e) Не был получен ответ на запрос открытия соединения.
  f) Параметр *DataSize* содержит значение от 510 до 1994, при этом удаленный узел, с которым открывается соединение, не поддерживает соединения Large_Forward_Open.
  g) Имеется ошибка в настройке локального IP-адреса.
  h) Произошла ошибка дублирования IP-адреса.
  i) Все TCP-соединения уже используются.
  j) Команда была выполнена при ошибке сервера BOOTP.

Информация о версии

При использовании модуля ЦПУ версии 1.10 или более поздней версии значение *Handle* не изменяется, даже если значение *Error* меняется на ИСТИНА. В случае версии 1.09 или более ранней версии значение *Handle* меняется на *0*.

CIPRead

Команда CIPRead использует явное сообщение класса 3 для чтения значения переменной в другом контроллере в сети CIP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CIPRead	Чтение переменной посредством явного сообщения класса 3	FB		CIPRead_instance(Execute, Handle, SrcDat, Size, DstDat, Done, Busy, Error, ErrorID, ErrorIDEx, RcvSize);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Handle	Дескриптор	Вход	Дескриптор, полученный при выполнении команды CIPOpen или CIPOpenWithDataSize	---	---	---
SrcDat	Имя читаемой переменной		Имя переменной в другом контроллере, значение которой нужно прочитать	Зависит от типа данных.		"
Size	Количество элементов для чтения		Количество элементов для чтения	0...8186*1		1
DstDat	Прочитанные данные	Вход-выход	Прочитанное значение	Зависит от типа данных.	---	---
RcvSize	Объем прочитанных данных	Выход	Объем прочитанных данных	0...8186*1	Байты	---

*1. Возможные значения для модулей ЦПУ NX1P2 и модулей ЦПУ серии NJ: *0...1988*.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Handle	Подробные сведения о структуре _sCIP_HANDLE см. в разделе *Функция* на стр. 2-1179.																			
SrcDat																				ОК
Size							ОК													
DstDat	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК
	Также можно указать перечисление, массив, структуру, член структуры или член объединения.*1																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RcvSize							OK													

*1. Нельзя указывать массив со строковыми данными (STRING).

Функция

Команда CIPRead производит чтение значения сетевой переменной, указанной параметром *SrcDat* (имя читаемой переменной), из другого контроллера в сети CIP. Другой контроллер указывается с помощью параметра *Handle*.
Прочитанные данные сохраняются в переменную *DstDat*.

Количество элементов, которые должны быть прочитаны, указывается параметром *Size*.
Если *SrcDat* является массивом, следует указать количество элементов массива, которые должны быть прочитаны. Если же *SrcDat* не является массивом, всегда следует указывать значение *1*.

Если для *Size* будет указано значение *0*, ничего не будет прочитано, независимо от того, является ли *SrcDat* массивом или нет.
После завершения операции чтения в переменную *RcvSize* (объем прочитанных данных) записывается количество байтов прочитанных данных.
Максимальный объем данных, которые могут быть прочитаны, зависит от команды, которая установила соединение, а также от типа читаемых данных. Сказанное отражено в таблице ниже.

Команда, установившая соединение	Тип читаемых данных	Максимальный объем данных, которые могут быть прочитаны [байт]
CIPOpen	Структура	1984
	STRING	1986
	Другой тип данных	1988
CIPOpenWithDataSize	Структура	Значение *DataSize* в команде CIPOpenWithDataSize - 10
	STRING	Значение *DataSize* в команде CIPOpenWithDataSize - 8
	Другой тип данных	Значение *DataSize* в команде CIPOpenWithDataSize - 6

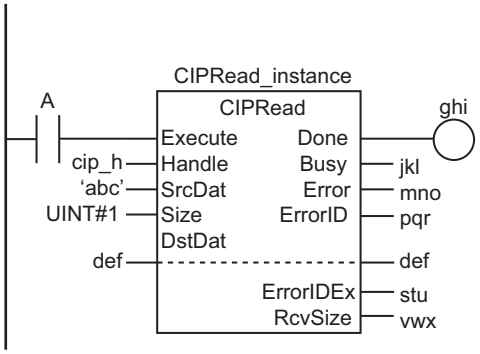
Для переменной *Handle* используется структурный тип данных _sCIP_HANDLE. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Handle	Дескриптор	Дескриптор	_sCIP_HANDLE	---	---	---
Handle	Дескриптор	Дескриптор	UDINT	Зависит от типа данных.	---	---

Если значение *ErrorID* = WORD#16#1C00, в переменную *ErrorIDEx* сохраняется код ошибки сообщения CIP.

В приведенном ниже примере считывается значение переменной *abc* в удаленном контроллере. Прочитанное значение сохраняется в переменную *def* в локальном контроллере. Количество элементов для чтения *Size* = UINT#1.
Тип данных переменных *abc* и *def*: SINT.
Размер значения типа SINT составляет один байт, поэтому значение объема прочитанных данных *vwx* равно UINT#1.

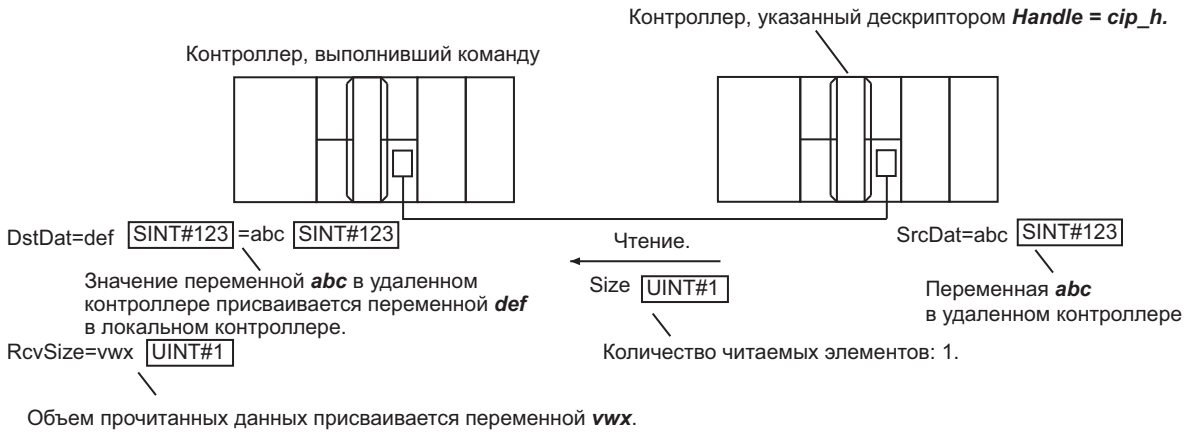
LD



ST

```
CIPRead_instance(A, cip_h, 'abc', UINT#1,
                 def, ghi, jkl, mno, pqr,
                 stu, vwx);
```

Значение переменной ***SrcDat*** в удаленном контроллере в сети CIP, указанном параметром ***Handle*** (дескриптор), присваивается переменной ***DstDat*** в локальном контроллере. Количество элементов, которые должны быть прочитаны, указывается в ***Size***. Объем прочитанных данных выводится в ***RcvSize***.

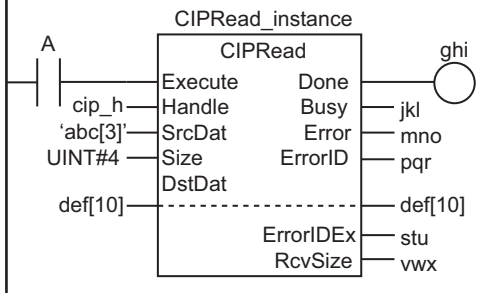


Чтение массивов

Для чтения данных массива в переменную *SrcDat* в качестве параметра следует передать элемент массива с указанием индекса. В переменную *DstDat* также следует передать в качестве параметра элемент массива с указанием индекса.

В приведенном ниже примере производится чтение содержимого четырех элементов переменной-массива abc[3]...abc[6] из удаленного контроллера. Результаты чтения записываются в элементы переменной-массива def[10]...def[13] в локальном контроллере.
Тип данных переменных *abc* и *def*: INT.
Размер значения типа INT составляет два байта, поэтому значение объема прочитанных данных *vwx* равно UINT#8.

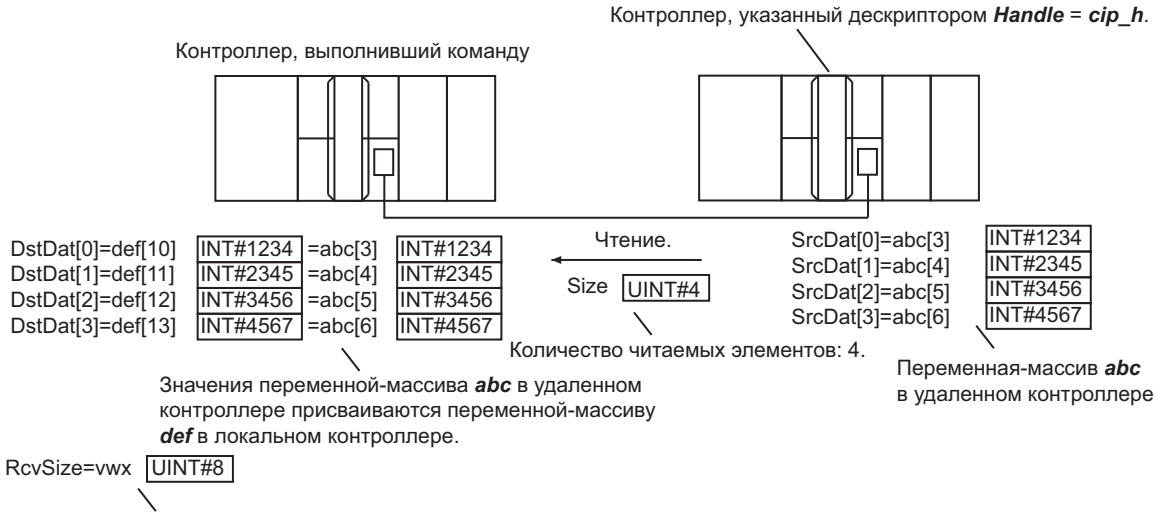
LD



ST

```
CIPRead_instance(A, cip_h, 'abc[3]', UINT#4,
                 def[10], ghi, jkl, mno, pqr,
                 stu, vwx);
```

Значения элементов переменной-массива ***abc[3]…abc[6]*** в удаленном контроллере присваиваются элементам переменной-массива ***def[10]…def[13]*** в локальном контроллере.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения об интерфейсе связи CIP см. в следующих документах.

- *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*
- *Серия CJ, модули EtherNet/IP — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W495)*

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Для получения значения дескриптора (*Handle*) перед выполнением этой команды следует выполнить команду CIPOpen или CIPOpenWithDataSize.
- Эту команду можно использовать для встроенного порта EtherNet/IP модуля ЦПУ серии NJ/NX или порта модуля интерфейса EtherNet/IP, подключенного к модулю ЦПУ серии NJ.
- Если читается значение переменной в контроллере OMRON, переменная должна быть открыта для доступа по сети («опубликована»). Это должно быть сделано заранее.
- Для чтения данных нельзя указать непосредственно значение адреса памяти, используемой для модулей серии CJ. Если нужно прочитать данные по конкретному адресу в памяти, используемой для модулей серии CJ, необходимо заранее назначить переменной требуемый адрес в памяти с помощью параметра «AT».
- Для записи данных нельзя указать непосредственно значение адреса локальной памяти, используемой для модулей серии CJ. Если нужно сохранить данные по конкретному адресу в памяти, используемой для модулей серии CJ, необходимо заранее назначить переменной *DstDat* требуемый адрес в памяти с помощью параметра «AT».
- В следующей таблице указаны символы, которые можно использовать в *SrcDat*.

Параметр	Требования
Максимальное количество байтов	127 байт
Кодировка	UTF-8
Применимые символы	Буквенно-цифровые символы (регистр не учитывается), однобайтовые символы катаканы, многобайтовые символы и «_» (символы подчеркивания)
Недопустимые текстовые строки	• Любая текстовая строка, начинающаяся с символа от 0 до 9 (коды символов от 16#30 до 16#39) ASCII. • Текстовая строка, состоящая только из одного символа «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, содержащая подряд два и более символов «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, начинающаяся с символа «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, заканчивающаяся символом «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, начинающаяся с «P_».

- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение *Size* находится за пределами допустимого диапазона.
  b) В параметр *SrcDat* передана недопустимая текстовая строка.
  c) Тип данных прочитанного значения не соответствует типу данных *DstDat*.
  d) Объем прочитанных данных превышает размер переменной *DstDat*.
  e) Для переменной *DstDat* был указан неподдерживаемый тип данных.
  f) Был возвращен ответ с ошибкой, которая определена в протоколе CIP.
  g) Значение *Handle.Handle* находится за пределами допустимого диапазона.
  h) Одновременно было выполнено более 32 команд, связанных с протоколом CIP.
  i) Превышено время ожидания для соединения, установленного с помощью команды CIPOpen или CIPOpenWithDataSize.

j) Размер *SrcDat* превосходит размер данных, который определен командой, установившей соединение, а также типом читаемых данных.

- Для этой команды в качестве дополнительного кода ошибки в переменной *ErrorIDEx* указывается код ошибки сообщения CIP. Значения кодов ошибки приведены в таблице ниже.

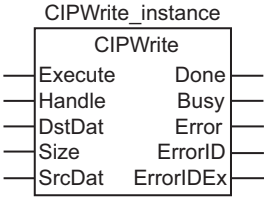
Значение	Ошибка
16#02000000	Нормальный обмен данными невозможен из-за высокой нагрузки на удаленном узле.
16#04000000	Для чтения указана переменная одного из следующих типов данных, который не предусмотрен в другом контроллере. • Базовый тип данных • Перечисление • Структура • Объединение • Массив
16#05000000	Для чтения указана переменная одного из следующих типов, который не предусмотрен в другом контроллере. • Перечислитель перечисления • Член структуры • Член объединения • Элемент массива
16#08000000	Запрошенная служба не поддерживается.
16#0C008010	Указанная для чтения переменная в данный момент загружается.
16#0C008011	
16#11000000	Значение *Size* превосходит объем данных, которые могут быть прочитаны в настоящее время.
16#1F000102	Для чтения указана переменная, чтение которой невозможно.
16#1F008007	Указанная переменная недоступна.
16#20008017	Указанная для чтения переменная не является массивом, при этом заданное количество элементов для чтения не равно 1.
16#20008018	Указанная для чтения переменная является массивом, при этом заданное количество элементов для чтения превышает фактическое количество элементов в массиве.
16#26000000	Указанная адресуемая переменная содержит только пустой символ (NULL).

Пример программы

См. *Пример программы* на стр. 2-1166 для команды CIPOpen.

CIPWrite

Команда CIPWrite использует явное сообщение класса 3 для записи значения в переменную в другом контроллере в сети CIP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CIPWrite	Запись переменной посредством явного сообщения класса 3	FB		CIPWrite_instance(Execute, Handle, DstDat, Size, SrcDat, Done, Busy, Error, ErrorID, ErrorIDEx);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Handle	Дескриптор	Вход	Дескриптор, полученный при выполнении команды CIPOpen или CIPOpenWithDataSize	---	---	---
DstDat	Имя записываемой переменной		Имя переменной в другом контроллере, в которую нужно записать значение	Зависит от типа данных.		"
Size	Количество записываемых элементов		Количество записываемых элементов	0...8 178*1		1
SrcDat	Исходные данные		Записываемое значение	Зависит от типа данных.		*2

*1. Возможные значения для модулей ЦПУ NX1P2 и модулей ЦПУ серии NJ: *0...1 980*.

*2. Если входной параметр будет опущен, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Handle	Подробные сведения о структуре _sCIP_HANDLE см. в разделе *Функция* на стр. 2-1185.																			
DstDat																				OK
Size							OK													
SrcDat	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать перечисление, массив*1, структуру, член структуры или член объединения.																			

*1. Нельзя указывать массив со строковыми данными (STRING).

Функция

Команда CIPWrite производит запись значения в сетевую переменную, указанную параметром *DstDat* (имя адресуемой переменной), в другом контроллере в сети CIP. Другой контроллер указывается с помощью параметра *Handle*.
Записывается содержимое переменной *SrcDat* (исходные данные).
Количество элементов, которые должны быть записаны, указывается параметром *Size*.
Если *DstDat* является массивом, следует указать количество элементов массива, в которые должна быть произведена запись.
Если же *DstDat* не является массивом, всегда следует указывать значение *1*.
Если для *Size* будет указано значение *0*, ничего не будет записано, независимо от того, является ли *DstDat* массивом или нет.

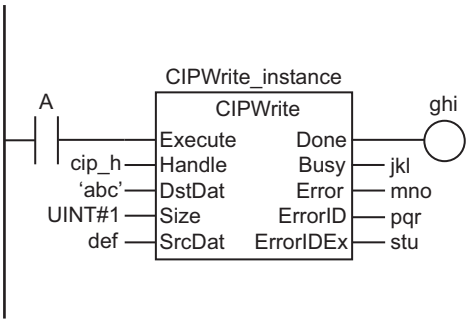
Для переменной *Handle* используется структурный тип данных _sCIP_HANDLE. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Handle	Дескриптор	Дескриптор	_sCIP_HANDLE	---	---	---
Handle	Дескриптор	Дескриптор	UDINT	Зависит от типа данных.	---	---

Если значение *ErrorID* = WORD#16#1C00, в переменную *ErrorIDEx* сохраняется код ошибки сообщения CIP.

В следующем примере значение переменной *def* из локального контроллера записывается в переменную *abc* в удаленном контроллере. Количество элементов для записи *Size* = UINT#1.

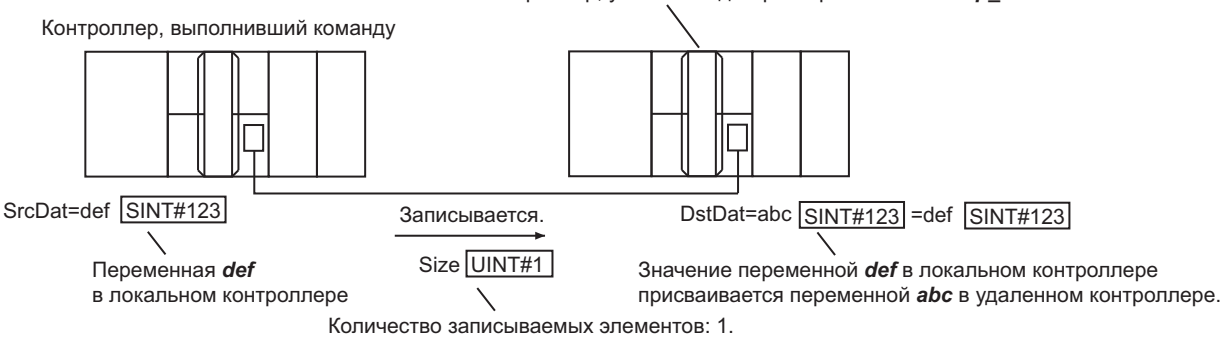
LD



ST

```
CIPWrite_instance(A, cip_h, 'abc', UINT#1,
                  def, ghi, jkl, mno, pqr, stu);
```

Значение переменной ***SrcDat*** в локальном контроллере присваивается переменной ***DstDat*** в удаленном контроллере в сети CIP, указанном параметром ***Handle*** (дескриптор).
Количество записываемых элементов указывается параметром ***Size***.

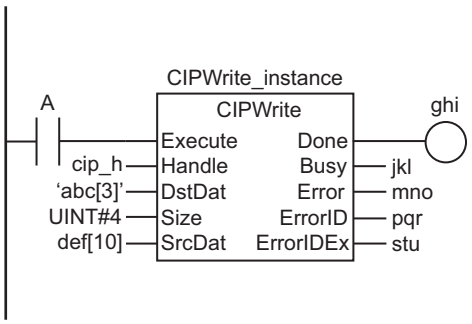


Запись массивов

Для записи данных массива в переменную *DstDat* в качестве параметра следует передать элемент массива с указанием индекса. В переменную *SrcDat* также следует передать в качестве параметра элемент массива с указанием индекса.

В следующем примере содержимое элементов переменной-массива def[10]...def [13] сохраняется в четыре элемента переменной-массива abc[3]...abc[6].

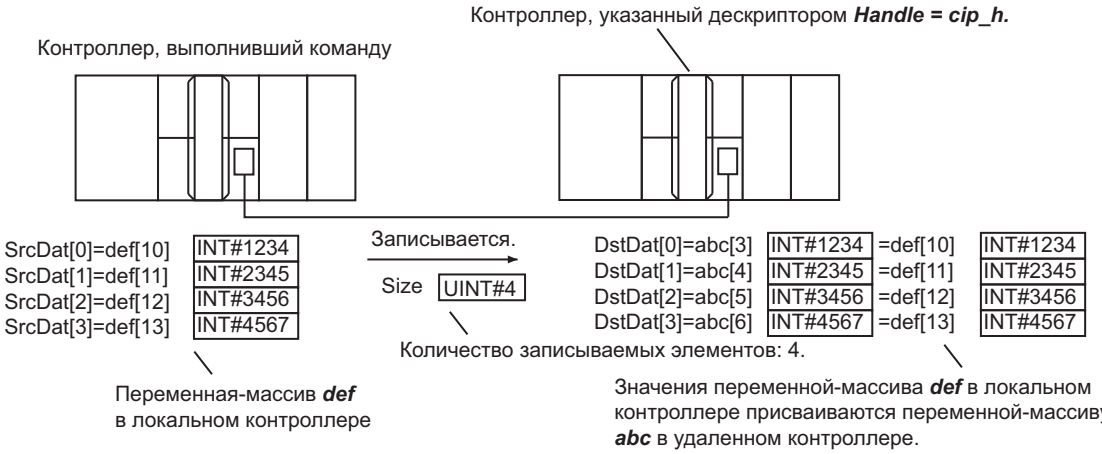
LD



ST

```
CIPWrite_instance(A, cip_h, 'abc[3]', UINT#4, def,
                  ghi[10], jkl, mno, pqr, stu);
```

Значения элементов переменной-массива ***def[10]…def[13]*** в локальном контроллере присваиваются элементам переменной-массива ***abc[3]…abc[6]*** в удаленном контроллере.



Максимальный объем записываемых данных

Максимальный объем данных, которые могут быть записаны, зависит от типа данных, а также от имени переменной, которое указано в *DstDat*. Сказанное отражено в следующей таблице.

Максимальный объем записываемых данных [байт] = базовый объем - размер имени переменной в *DstDat*

Параметр в формуле выше	Значение
Базовый объем	Соединения, установленные с помощью команды CIPOpen • Тип данных переменной, указанной для *DstDat* — структура: 1 984 байт • Тип данных переменной, указанной для *DstDat* — STRING: 1 986 байт • Другие типы данных: 1 988 байт
	Соединения, установленные с помощью команды CIPOpenWithDataSize • Если тип данных переменной, указанной для *DstDat* — структура, используйте следующую формулу: Базовый объем (**байт**) = значение *DataSize* в команде CIPOpenWithDataSize - 10 • Если тип данных переменной, указанной для *DstDat* — STRING, используйте следующую формулу: Базовый объем (**байт**) = значение *DataSize* в команде CIPOpenWithDataSize - 8 • Для других типов данных используйте следующую формулу: Базовый объем (**байт**) = значение *DataSize* в команде CIPOpenWithDataSize - 6
Размер имени переменной в *DstDat*	• Размер имени переменной рассчитывается как общее количество байтов для символов ASCII на всех уровнях структуры плюс удвоенное количество уровней. • Если количество байтов символов ASCII на некотором уровне нечетное, следует добавить 1. • Если некоторый уровень в структуре является массивом, следует добавить количество измерений в массиве, помноженное на 4. • Точки и запятые в структуре и массивах в размер имени переменной не включаются. Пример 1. Имя переменной в *DstDat*: «aaa.bbbbb[1,2,3].cc» • Размер текстовой строки "aaa" на первом уровне составляет 3 байта. Это нечетное число, поэтому добавляем к нему 1 и получаем 4 байта. • "bbbbb" в "bbbbb[1,2,3]" на втором уровне — это текстовая строка размером в 5 байтов. Это нечетное число, поэтому добавляем к нему 1 и получаем 6 байтов. • Кроме того, "bbbbb[1,2,3]" — это трехмерный массив, поэтому также добавляем 12 (3 x 4) и получаем в итоге 18 байтов. • Размер текстовой строки "cc" на третьем уровне составляет 2 байта. Это четное число, поэтому при вычислении используется 2 байта. • Умножим количество уровней 3 на 2 и получим 6. Теперь сложим все полученные значения: 6 + 4 байта для первого уровня + 18 байтов для второго уровня + 2 байта для третьего уровня. Итого, размер имени переменной составляет 30 байтов. Пример 2. Имя переменной в *DstDat*: «val» • Размер текстовой строки "val" на первом уровне составляет 3 байта. Это нечетное число, поэтому добавляем к нему 1 и получаем 4 байта. • Умножим количество уровней 1 на 2 и получим 2. Следовательно, размер имени переменной составляет 2 + 4 = 6 байтов. Пример 3. Имя переменной в *DstDat*: «array[8]» • Размер текстовой строки "array" на первом уровне составляет 5 байтов. Это нечетное число, поэтому добавляем к нему 1 и получаем 6 байтов. • Это одномерный массив. Поэтому также нужно добавить 4 (результат умножения 1 на 4). • Умножим количество уровней 1 на 2 и получим 2. Следовательно, размер имени переменной составляет 2 + 6 + 4 = 12 байтов.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения об интерфейсе связи CIP см. в следующих документах.

- *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*
- *Серия CJ, модули EtherNet/IP — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W495)*

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Для получения значения дескриптора (*Handle*) перед выполнением этой команды следует выполнить команду CIPOpen или CIPOpenWithDataSize.
- Всегда используйте переменную в качестве входного параметра, передаваемого в *SrcDat*. При передаче константы произойдет ошибка сборки.
- Если параметр *SrcDat* является перечислением, в него нельзя напрямую передавать перечислитель. В случае непосредственной передачи перечислителя произойдет ошибка сборки.
- Эту команду можно использовать для встроенного порта EtherNet/IP модуля ЦПУ серии NJ/NX или порта модуля интерфейса EtherNet/IP, подключенного к модулю ЦПУ серии NJ.
- Если переменная записывается в контроллер OMRON, она должна быть открыта для доступа по сети («опубликована»). Это должно быть сделано заранее.
- Для записи данных нельзя указать непосредственно значение адреса памяти, используемой для модулей серии CJ. Если нужно записать данные по конкретному адресу в памяти, используемой для модулей серии CJ, необходимо заранее назначить переменной требуемый адрес в памяти с помощью параметра «AT».
- Нельзя указать непосредственно значение адреса локальной памяти, используемой для модулей серии CJ. Если нужно получить данные по конкретному адресу в памяти, используемой для модулей серии CJ, необходимо заранее назначить переменной *SrcDat* требуемый адрес в памяти с помощью параметра «AT».
- В следующей таблице указаны символы, которые можно использовать в *DstDat*.

Параметр	Требования
Максимальное количество байтов	127 байт
Кодировка	UTF-8
Применимые символы	Буквенно-цифровые символы (регистр не учитывается), однобайтовые символы катаканы, многобайтовые символы и «_» (символы подчеркивания)
Недопустимые текстовые строки	• Любая текстовая строка, начинающаяся с символа от 0 до 9 (коды символов от 16#30 до 16#39) ASCII. • Текстовая строка, состоящая только из одного символа «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, содержащая подряд два и более символов «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, начинающаяся с символа «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, заканчивающаяся символом «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, начинающаяся с «P_».

- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение *Size* находится за пределами допустимого диапазона.
  b) В параметр *DstDat* передана недопустимая текстовая строка.
  c) Значение *Size* превышает размер переменной *SrcDat*.
  d) Для переменной *SrcDat* был указан неподдерживаемый тип данных.
  e) Был возвращен ответ с ошибкой, которая определена в протоколе CIP.
  f) Значение *Handle.Handle* находится за пределами допустимого диапазона.
  g) Одновременно было выполнено более 32 команд, связанных с протоколом CIP.
  h) Превышено время ожидания для соединения, установленного с помощью команды CIPOpen или CIPOpenWithDataSize.
  i) Размер имени переменной в *DstDat* и размер значения в *SrcDat* в сумме превосходят базовый объем, т. е. размер данных, который определен командой, установившей соединение, а также типом записываемых данных.
- Для этой команды в качестве дополнительного кода ошибки в переменной *ErrorIDEx* указывается код ошибки сообщения CIP. Значения кодов ошибки приведены в таблице ниже.

Значение	Ошибка
16#02000000	Нормальный обмен данными невозможен из-за высокой нагрузки на удаленном узле.
16#04000000	Для чтения указана переменная одного из следующих типов данных, который не предусмотрен в другом контроллере. • Базовый тип данных • Перечисление • Структура • Объединение • Массив
16#05000000	Для чтения указана переменная одного из следующих типов, который не предусмотрен в другом контроллере. • Перечислитель перечисления • Член структуры • Член объединения • Элемент массива
16#08000000	Запрошенная служба не поддерживается.
16#0C008010	Указанная для чтения переменная в данный момент загружается.
16#0C008011	
16#1F000102	• Указанная адресуемая переменная имеет атрибут Constant (Константа), поэтому запись в нее невозможна. • Записываемые данные не соответствуют количеству записываемых элементов.
16#1F008007	Указанная переменная недоступна.

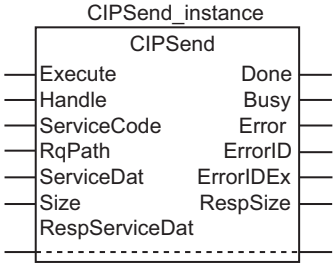
Значение	Ошибка
16#20008017	Указанная адресуемая переменная не является массивом, при этом заданное количество элементов для записи не равно 1.
16#20008018	Указанная адресуемая переменная является массивом, при этом заданное количество элементов для записи превышает фактическое количество элементов в массиве.
16#20008022	Тип данных переменных сервера отличается от типа данных переменных клиента.
16#20008028	• Указанная адресуемая переменная является перечислением, при этом записываемое значение не является значением перечислителя. • Для указанной адресуемой переменной задан атрибут диапазона значений, при этом записываемое значение находится вне этого диапазона.
16#26000000	Указанная адресуемая переменная содержит только пустой символ (NULL).

Пример программы

См. *Пример программы* на стр. 2-1166 для команды CIPOpen.

CIPSend

Команда CIPSend передает сообщение класса 3 по протоколу CIP указанному устройству в сети CIP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CIPSend	Передача явного сообщения класса 3	FB		CIPSend_instance(Execute, Handle, ServiceCode, RqPath, ServiceDat, Size, RespServiceDat, Done, Busy, Error, ErrorID, ErrorIDEx, RespSize);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Handle	Дескриптор	Вход	Дескриптор, полученный при выполнении команды CIPOpen или CIPOpenWithDataSize	---	---	---
ServiceCode	Код службы		Код службы	Зависит от типа данных.		
RqPath	Путь запроса		Путь запроса	---		
ServiceDat	Данные службы		Передаваемые данные службы	Зависит от типа данных.		*1
Size	Количество элементов для передачи		Количество элементов для передачи			1
RespServiceDat	Данные ответа	Вход-выход	Данные ответа	Зависит от типа данных.	---	---
RespSize	Размер ответа	Выход	Объем данных ответа	Зависит от типа данных.	Байты	---

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Handle	Подробные сведения о структуре _sCIP_HANDLE см. в разделе *Функция* на стр. 2-1192.																			
ServiceCode		OK																		

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RqPath	Структура _sREQUEST_PATH или _sREQUEST_PATH_EX*1. Дополнительные сведения см. в разделе *Тип данных переменной RqPath* на стр. 2-1193.																			
ServiceDat		ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
	Также можно указать массив, член структуры или член объединения.																			
Size							ОК													
RespServiceDat		ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
	Также можно указать массив, член структуры или член объединения.																			
RespSize							ОК													

*1. Чтобы можно было указать тип _sREQUEST_PATH_EX, требуется модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Функция

Команда CIPSend передает данные службы *ServiceDat* для службы, указанной кодом службы *ServiceCode*, в виде явного сообщения класса 3.
Адресат данных указывается с помощью дескриптора *Handle*.
В параметре *RqPath* указывается путь запроса.

В параметре *Size* указывается количество элементов, которые должны быть переданы.
Если *ServiceDat* является массивом, следует указать количество элементов массива, которые должны быть переданы.
Если же *ServiceDat* не является массивом, всегда следует указывать значение *1*.
Если передавать данные службы не требуется, *Size* можно задать равным *0*.

Принятые в ответ данные сохраняются в переменную *RespServiceDat*. Количество байтов данных ответа записывается в переменную *RespSize*.

Для переменной *Handle* используется структурный тип данных _sCIP_HANDLE. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Handle	Дескриптор	Дескриптор	_sCIP_HANDLE	---	---	---
Handle	Дескриптор	Дескриптор	UDINT	Зависит от типа данных.	---	---

Для параметров *ClassIDLogicalFormat*, *InstanceIDLogicalFormat* и *AttributeIDLogicalFormat* используется перечислимый тип данных _eCIP_LOGICAL_FORMAT.
Значения перечислителей перечислимого типа _eCIP_LOGICAL_FORMAT приведены в таблице ниже:

Перечислитель	Значение
_8BIT	8 бит
_16BIT	16 бит
_32BIT	32 бит

Если значение *ErrorID* = WORD#16#1C00, в переменную *ErrorIDEx* сохраняется код ошибки сообщения CIP.
Значение кода ошибки в *ErrorIDEx* и его смысл зависят от конкретного удаленного узла. См. руководство по соответствующему удаленному узлу.

Тип данных переменной *RqPath*

Для переменной *RqPath* используется структурный тип данных _sREQUEST_PATH или _sREQUEST_PATH_EX.
Как правило, используется тип _sREQUEST_PATH.
При указании какого-либо размера логического формата следует использовать тип _sREQUEST_PATH_EX.

● Тип данных _sREQUEST_PATH

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
RqPath	Путь запроса	Путь запроса	_sREQUEST_PATH	---	---	---
ClassID	Идентификатор класса	Идентификатор класса	UINT	Зависит от типа данных.	---	0
InstanceID	Идентификатор экземпляра	Идентификатор экземпляра	UINT			0
isAttributeID	Использование атрибута	ИСТИНА: используется идентификатор атрибута. ЛОЖЬ: идентификатор атрибута не используется.	BOOL			ЛОЖЬ
AttributeID	Идентификатор атрибута	Идентификатор атрибута	UINT			0

Примечание Размер логического формата каждого идентификатора в типе _sREQUEST_PATH составляет 16 бит.

● Тип данных _sREQUEST_PATH_EX

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
RqPath	Путь запроса	Путь запроса	_sREQUEST_PATH_EX	---	---	---
ClassIDLogicalFormat	Логический формат идентификатора класса	Размер данных идентификатора класса	_eCIP_LOGICAL_FORMAT	Зависит от типа данных.	---	_8BIT
ClassID	Идентификатор класса	Идентификатор класса	UDINT			0
InstanceIDLogicalFormat	Логический формат идентификатора экземпляра	Размер данных идентификатора экземпляра	_eCIP_LOGICAL_FORMAT			_8BIT
InstanceID	Идентификатор экземпляра	Идентификатор экземпляра	UDINT			0
isAttributeID	Использование атрибута	ИСТИНА: используется идентификатор атрибута. ЛОЖЬ: идентификатор атрибута не используется.	BOOL			ЛОЖЬ
AttributeIDLogicalFormat	Логический формат идентификатора атрибута	Размер данных идентификатора атрибута	_eCIP_LOGICAL_FORMAT			_8BIT
AttributeID	Идентификатор атрибута	Идентификатор атрибута	UDINT			0

Передача и прием массивов

Если переменная *ServiceDat* или *RespServiceDat* является массивом, в нее в качестве параметра следует передать элемент массива с указанием индекса.

Максимальный объем читаемых/записываемых данных

Максимальный объем данных, которые могут быть прочитаны, зависит от того, с помощью какой команды было установлено соединение: CIPOpen или CIPOpenWithDataSize. Это отражено в таблице ниже.

Команда, открывшая соединение	Максимальный объем данных, которые могут быть прочитаны
CIPOpen	1 990 байт
CIPOpenWithDataSize	От сервера может быть получено до 8188 байт данных ответа.*1

*1. Максимальный объем для модулей ЦПУ NX1P2 и модулей ЦПУ серии NJ: *1990*.

Максимальный объем данных, которые могут быть записаны, зависит от того, имеется ли атрибут пути запроса, а также от команды, которая установила соединение. Сказанное отражено в таблице ниже.

Максимальный объем записываемых данных [байт] = базовый объем - объем, зависящий от использования атрибута

Параметр в формуле выше	Значение
Базовый объем	• Соединение установлено с помощью команды CIPOpen: 1 992 байт • Соединение установлено с помощью команды CIPOpenWithDataSize: значение *DataSize* для команды CIPOpenWithDataSize - 2
Использование атрибута*1	• Идентификатор атрибута используется: 14 байт • Идентификатор атрибута не используется: 10 байт

*1. В случае модуля ЦПУ с версией модуля 1.10 или более ранней или Sysmac Studio версии 1.14 или ниже применяются следующие значения:
Идентификатор атрибута используется: 12 байт
Идентификатор атрибута не используется: 8 байт

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения об интерфейсе связи CIP см. в следующих документах.

- *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*
- *Серия CJ, модули EtherNet/IP — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W495)*

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Для получения значения дескриптора (*Handle*) перед выполнением этой команды следует выполнить команду CIPOpen или CIPOpenWithDataSize.

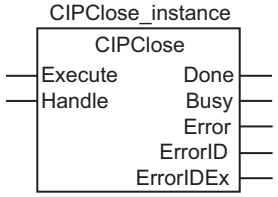
- Всегда используйте переменную в качестве входного параметра, передаваемого в *ServiceDat*. При передаче константы произойдет ошибка сборки.
- Эту команду можно использовать для встроенного порта EtherNet/IP модуля ЦПУ серии NJ/NX или порта модуля интерфейса EtherNet/IP, подключенного к модулю ЦПУ серии NJ.
- Если переменная записывается в контроллер OMRON, она должна быть открыта для доступа по сети («опубликована»). Это должно быть сделано заранее.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для параметра *RqPath.ClassIDLogicalFormat* или *RqPath.AttributeIDLogicalFormat* указано значение, выходящее за пределы допустимого диапазона.
  b) Возникло несоответствие между следующими двумя переменными: размер, указанный в *RqPath.ClassIDLogicalFormat*, и размер значения *RqPath.ClassID*; размер, указанный в *RqPath.InstanceIDLogicalFormat*, и размер значения *RqPath.InstanceID*; или размер, указанный в *RqPath.AttributeIDLogicalFormat*, и размер значения *RqPath.AttributeID*.
  c) Значение *Size* превышает размер записываемых данных.
  d) Значение *Size* превышает размер переменной *ServiceDat*.
  e) Значение *RespSize* превышает размер переменной *RespServiceDat*.
  f) Для переменной *ServiceDat* был указан неподдерживаемый тип данных.
  g) Для переменной *RespServiceDat* был указан неподдерживаемый тип данных.
  h) Тип данных переменной, указанной для пути запроса *RqPath*, отличается от _sREQUEST_PATH и _sREQUEST_PATH_EX.
  i) Был возвращен ответ с ошибкой, которая определена в протоколе CIP.
  j) Значение *Handle.Handle* находится за пределами допустимого диапазона.
  k) Одновременно было выполнено более 32 команд, связанных с протоколом CIP.
  l) Превышено время ожидания для соединения, установленного с помощью команды CIPOpen или CIPOpenWithDataSize.
  m) Размер пути запроса в *RqPath* и размер данных службы в *ServiceDat* в сумме превосходят размер данных, который определен командой, установившей соединение.

Пример программы

См. *Пример программы* на стр. 2-1166 для команды CIPOpen.

CIPClose

Команда CIPClose закрывает соединение класса 3 по протоколу CIP для указанного дескриптора.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CIPClose	Закрытие CIP-соединения класса 3	FB		CIPClose_instance(Execute, Handle, Done, Busy, Error, ErrorID, ErrorIDEx);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Handle	Дескриптор	Вход	Дескриптор, полученный при выполнении команды CIPOpen или CIPOpenWithDataSize	---	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Handle	Подробные сведения о структуре _sCIP_HANDLE см. в разделе *Функция* на стр. 2-1197.																			

Функция

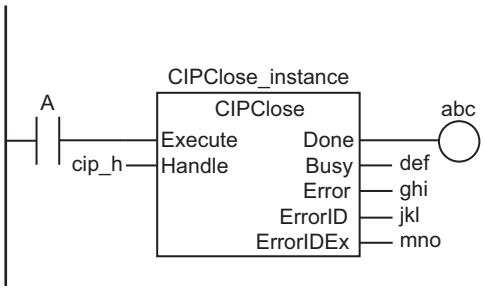
Команда CIPClose закрывает соединение класса 3 по протоколу CIP, указанное дескриптором *Handle*.

Для переменной *Handle* используется структурный тип данных _sCIP_HANDLE. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Handle	Дескриптор	Дескриптор	_sCIP_HANDLE	---	---	---
Handle	Дескриптор	Дескриптор	UDINT	Зависит от типа данных.	---	---

Пример программы представлен на рисунке ниже. Команда CIPClose закрывает соединение класса 3 по протоколу CIP, указанное дескриптором *Handle* (= cip_h).

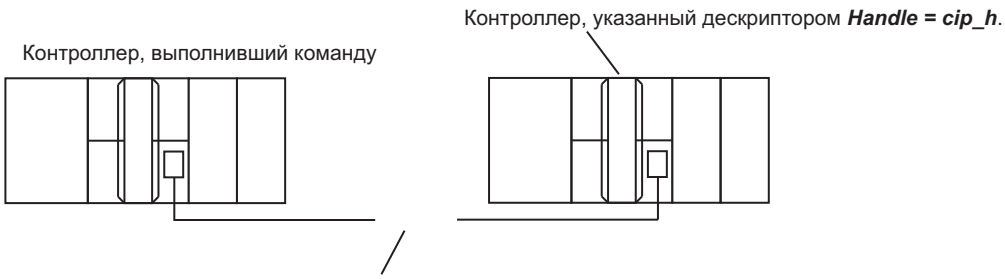
LD



ST

```
CIPClose_instance(A, cip_h, abc, def,
                  ghi, jkl, mno);
```

Команда CIPClose закрывает CIP-соединение класса 3, указанное с помощь ***Handle***.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения об интерфейсе связи CIP см. в следующих документах.

- *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*
- *Серия CJ, модули EtherNet/IP — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W495)*

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По

завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.

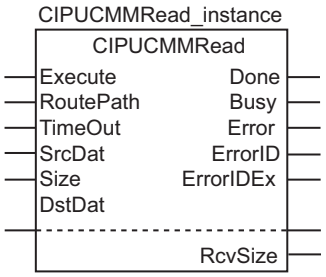
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- В переменной *Handle* следует указать дескриптор, который был получен с помощью команды CIPOpen или CIPOpenWithDataSize.
- Эту команду можно использовать для встроенного порта EtherNet/IP модуля ЦПУ серии NJ/NX или порта модуля интерфейса EtherNet/IP, подключенного к модулю ЦПУ серии NJ.
- Эта команда не использует *ErrorIDEx*.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение *Handle.Handle* находится за пределами допустимого диапазона.
  b) Одновременно было выполнено более 32 команд, связанных с протоколом CIP.

Пример программы

См. *Пример программы* на стр. 2-1166 для команды CIPOpen.

CIPUCMMRead

Команда CIPUCMMRead использует явное сообщение UCMM для чтения значения переменной в другом контроллере в указанной сети CIP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CIPUCMMRead	Чтение переменной посредством явного сообщения UCMM	FB		CIPUCMMRead_instance(Execute , RoutePath, TimeOut, SrcDat, Size, DstDat, Done, Busy, Error, ErrorID, ErrorIDEx, RcvSize);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
RoutePath	Путь маршрута	Вход	Путь маршрута	Зависит от типа данных.	---	---
TimeOut	Время ожидания		Время ожидания	1…65535	0,1 с	20 (2 с)
SrcDat	Имя читаемой переменной		Имя переменной в другом контроллере, значение которой нужно прочитать	Зависит от типа данных.	---	"
Size	Количество элементов для чтения		Количество элементов для чтения	0…496		1
DstDat	Прочитанные данные	Вход-выход	Прочитанное значение	Зависит от типа данных.	---	---
RcvSize	Объем прочитанных данных	Выход	Объем прочитанных данных	0…496	Байты	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RoutePath																				OK
TimeOut							OK													
SrcDat																				OK
Size							OK													
DstDat	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать перечисление, массив, структуру, член структуры или член объединения.*1																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RcvSize							OK													

*1. Нельзя указывать массив со строковыми данными (STRING).

Функция

Команда CIPUCMMRead производит чтение значения сетевой переменной, указанной параметром *SrcDat* (имя читаемой переменной), из другого контроллера в сети CIP. Другой контроллер указывается с помощью параметра *RoutePath* (путь маршрута).
Прочитанные данные сохраняются в переменную *DstDat*.

Количество элементов, которые должны быть прочитаны, указывается параметром *Size*.
Если *SrcDat* является массивом, следует указать количество элементов массива, которые должны быть прочитаны.
Если же *SrcDat* не является массивом, всегда следует указывать значение *1*.
Если для *Size* будет указано значение *0*, ничего не будет прочитано, независимо от того, является ли *SrcDat* массивом или нет.

После завершения операции чтения в переменную *RcvSize* (объем прочитанных данных) записывается количество байтов прочитанных данных. Максимальный объем данных, которые могут быть прочитаны, зависит от типа данных переменной следующим образом:

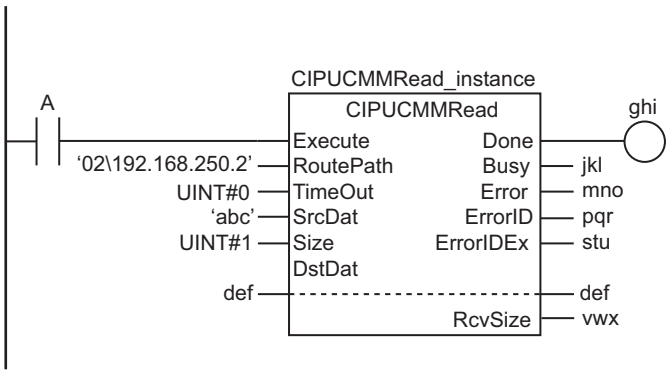
- Структура: 492 байт
- STRING: 494 байт
- Другие типы данных: 496 байт

Параметр *TimeOut* задает время ожидания. Если ответ не возвращается в течение времени ожидания, считается, что произошел сбой связи.

Если значение *ErrorID* = WORD#16#1C00, в переменную *ErrorIDEx* сохраняется код ошибки сообщения CIP.

В приведенном ниже примере считывается значение переменной *abc* в удаленном контроллере. Прочитанное значение сохраняется в переменную *def* в локальном контроллере.
Количество элементов для чтения *Size* = UINT#1.
Тип данных переменных *abc* и *def*: SINT.
Размер значения типа SINT составляет один байт, поэтому значение объема прочитанных данных *vwx* равно UINT#1.

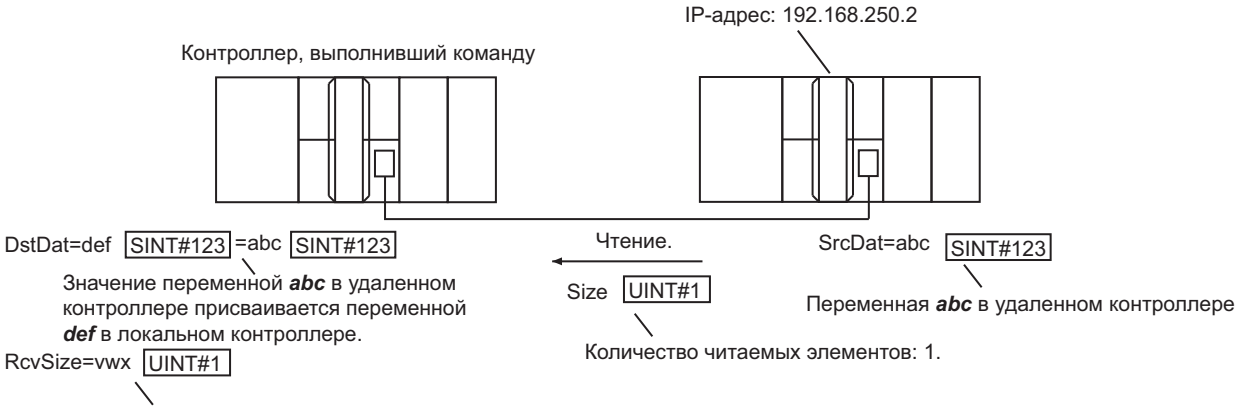
LD



ST

```
CIPUCMMRead_instance(A, '2\192.168.250.2',
                     UINT#0, 'abc', UINT#1,
                     def, ghi, jkl, mno, pqr,
                     stu, vwx);
```

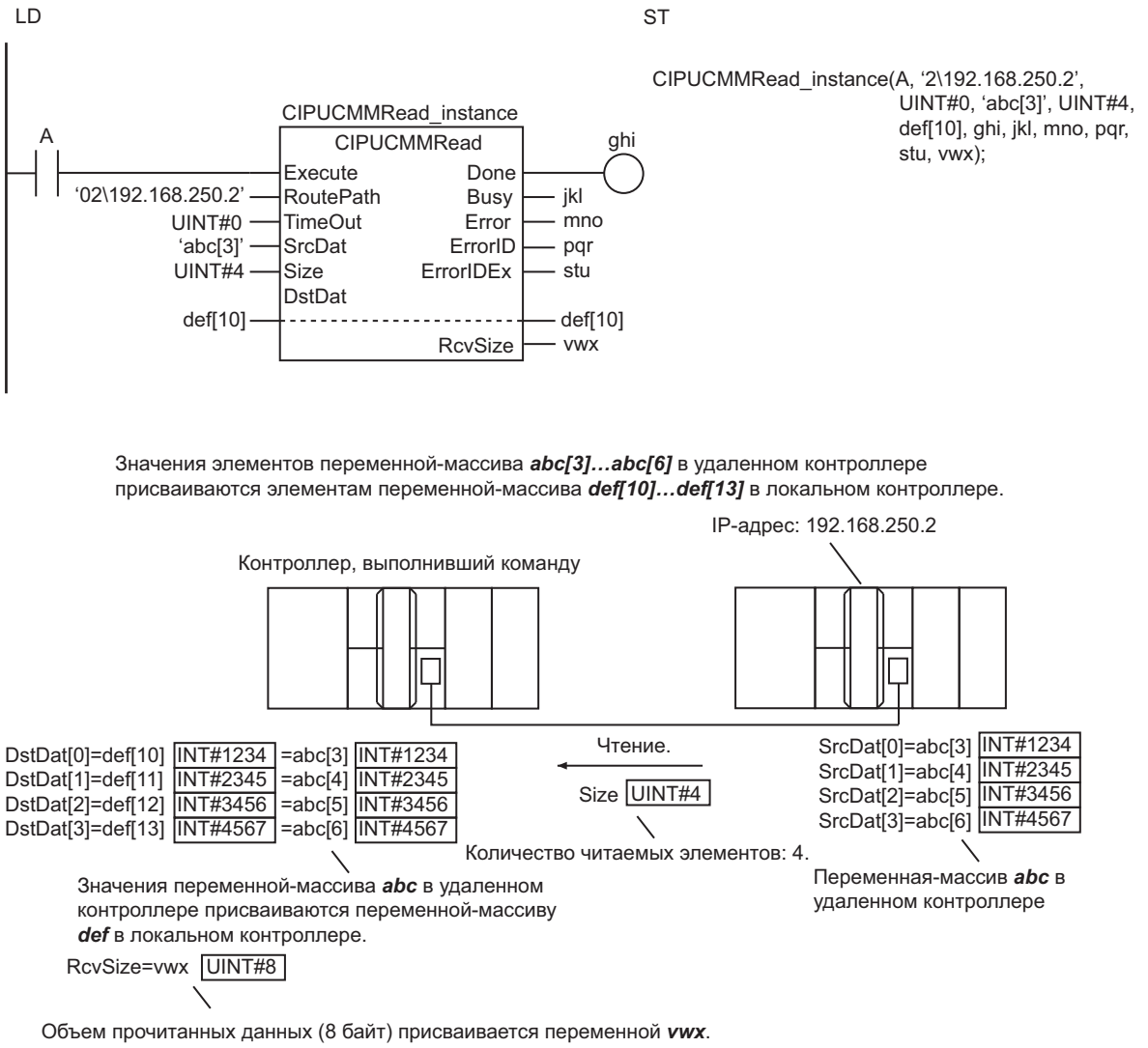
Значение переменной ***SrcDat*** в удаленном контроллере в сети CIP, указанном параметром ***RoutePath*** (путь маршрута), присваивается переменной ***DstDat*** в локальном контроллере.
Количество элементов, которые должны быть прочитаны, указывается в ***Size***.
Объем прочитанных данных выводится в ***RcvSize***.



Чтение массивов

Для чтения данных массива в переменную *SrcDat* в качестве параметра следует передать элемент массива с указанием индекса. В переменную *DstDat* также следует передать в качестве параметра элемент массива с указанием индекса.

В приведенном ниже примере производится чтение содержимого четырех элементов переменной-массива abc[3]...abc[6] из удаленного контроллера. Результаты чтения записываются в элементы переменной-массива def[10]...def[13] в локальном контроллере.
Тип данных переменных *abc* и *def*: INT.
Размер значения типа INT составляет два байта, поэтому значение объема прочитанных данных *vwx* равно UINT#8.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения об интерфейсе связи CIP см. в следующих документах.

- *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*
- *Серия CJ, модули EtherNet/IP — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W495)*

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Если читается значение переменной в контроллере OMRON, переменная должна быть открыта для доступа по сети («опубликована»). Это должно быть сделано заранее.
- Для чтения данных нельзя указать непосредственно значение адреса памяти, используемой для модулей серии CJ. Если нужно прочитать данные по конкретному адресу в памяти, используемой для модулей серии CJ, необходимо заранее назначить переменной требуемый адрес в памяти с помощью параметра «AT».
- Для записи данных нельзя указать непосредственно значение адреса локальной памяти, используемой для модулей серии CJ. Если нужно сохранить данные по конкретному адресу в памяти, используемой для модулей серии CJ, необходимо заранее назначить переменной *DstDat* требуемый адрес в памяти с помощью параметра «AT».
- В следующей таблице указаны символы, которые можно использовать в *SrcDat*.

Параметр	Требование
Максимальное количество байтов	127 байт
Кодировка	UTF-8
Применимые символы	Буквенно-цифровые символы (регистр не учитывается), однобайтовые символы катаканы, многобайтовые символы и «_» (символы подчеркивания)
Недопустимые текстовые строки	• Любая текстовая строка, начинающаяся с символа от 0 до 9 (коды символов от 16#30 до 16#39) ASCII. • Текстовая строка, состоящая только из одного символа «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, содержащая подряд два и более символов «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, начинающаяся с символа «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, заканчивающаяся символом «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, начинающаяся с «P_».

- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение *TimeOut* находится за пределами допустимого диапазона.
  b) Значение *Size* находится за пределами допустимого диапазона.
  c) В параметр *SrcDat* передана недопустимая текстовая строка.
  d) Тип данных прочитанного значения не соответствует типу данных *DstDat*.
  e) Объем прочитанных данных превышает размер переменной *DstDat*.
  f) Для переменной *DstDat* был указан неподдерживаемый тип данных.
  g) Был возвращен ответ с ошибкой, которая определена в протоколе CIP.
  h) В параметр *RoutePath* передана недопустимая текстовая строка.
  i) Одновременно было выполнено более 32 команд, связанных с протоколом CIP.
  j) Ответ не был получен в течение заданного времени ожидания.
  k) Имеется ошибка в настройке локального IP-адреса.
  l) Команда была выполнена при ошибке сервера BOOTP.

m) Произошла ошибка дублирования IP-адреса.

- Для этой команды в качестве дополнительного кода ошибки в переменной *ErrorIDEx* указывается код ошибки сообщения CIP. Значения кодов ошибки приведены в таблице ниже.

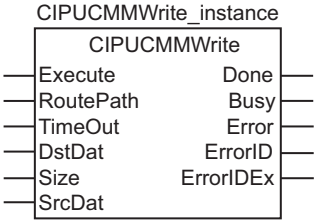
Значение	Ошибка
16#02000000	Нормальный обмен данными невозможен из-за высокой нагрузки на удаленном узле.
16#04000000	Для чтения указана переменная одного из следующих типов данных, который не предусмотрен в другом контроллере. • Базовый тип данных • Перечисление • Структура • Объединение • Массив
16#05000000	Для чтения указана переменная одного из следующих типов, который не предусмотрен в другом контроллере. • Перечислитель перечисления • Член структуры • Член объединения • Элемент массива
16#08000000	Запрошенная служба не поддерживается.
16#0C008010	Указанная для чтения переменная в данный момент загружается.
16#0C008011	
16#11000000	Значение *Size* превосходит объем данных, которые могут быть прочитаны в настоящее время.
16#1F000102	Для чтения указана переменная, чтение которой невозможно.
16#1F008007	Указанная переменная недоступна.
16#20008017	Указанная для чтения переменная не является массивом, при этом заданное количество элементов для чтения не равно 1.
16#20008018	Указанная для чтения переменная является массивом, при этом заданное количество элементов для чтения превышает фактическое количество элементов в массиве.
16#26000000	Указанная адресуемая переменная содержит только пустой символ (NULL).

Пример программы

См. *Пример программы* на стр. 2-1220 для команды CIPUCMMSend.

CIPUCMMWrite

Команда CIPUCMMWrite использует явное сообщение UCMM для записи значения в переменную в другом контроллере в сети CIP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CIPUCMMWrite	Запись переменной посредством явного сообщения UCMM	FB		CIPUCMMWrite_instance(Execute , RoutePath, TimeOut, DstDat, Size, SrcDat, Done, Busy, Error, ErrorID, ErrorIDEx);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
RoutePath	Путь маршрута	Вход	Путь маршрута	Зависит от типа данных.	---	---
TimeOut	Время ожидания		Время ожидания	1…65535	0,1 с	20 (2 с)
DstDat	Имя записываемой переменной		Имя переменной в другом контроллере, в которую нужно записать значение	Зависит от типа данных.	---	"
Size	Количество записываемых элементов		Количество записываемых элементов	0…488		1
SrcDat	Исходные данные		Записываемое значение	Зависит от типа данных.		*1

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RoutePath																				OK
TimeOut							OK													
DstDat																				OK
Size							OK													
SrcDat	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать перечисление, массив*1, структуру, член структуры или член объединения.																			

*1. Нельзя указывать массив со строковыми данными (STRING).

Функция

Команда CIPUCMMWrite производит запись значения в сетевую переменную, указанную параметром *DstDat* (имя адресуемой переменной), в другом контроллере в сети CIP. Другой контроллер указывается с помощью параметра *RoutePath* (путь маршрута).
Записывается содержимое переменной *SrcDat* (исходные данные).

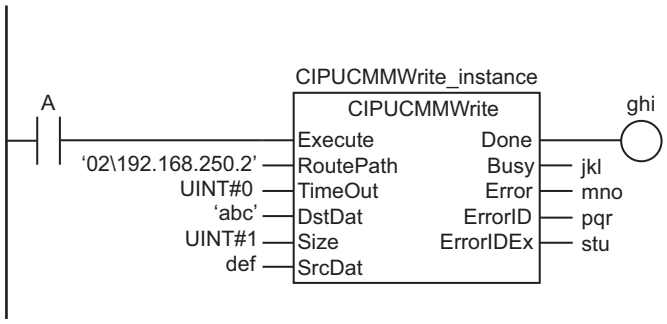
Количество элементов, которые должны быть записаны, указывается параметром *Size*.
Если *DstDat* является массивом, следует указать количество элементов массива, в которые должна быть произведена запись.
Если же *DstDat* не является массивом, всегда следует указывать значение *1*.
Если для *Size* будет указано значение *0*, ничего не будет записано, независимо от того, является ли *DstDat* массивом или нет.

Параметр *TimeOut* задает время ожидания. Если ответ не возвращается в течение времени ожидания, считается, что произошел сбой связи.

Если значение *ErrorID* = WORD#16#1C00, в переменную *ErrorIDEx* сохраняется код ошибки сообщения CIP.

В следующем примере значение переменной *def* из локального контроллера записывается в переменную *abc* в удаленном контроллере. Количество элементов для записи *Size* = UINT#1.

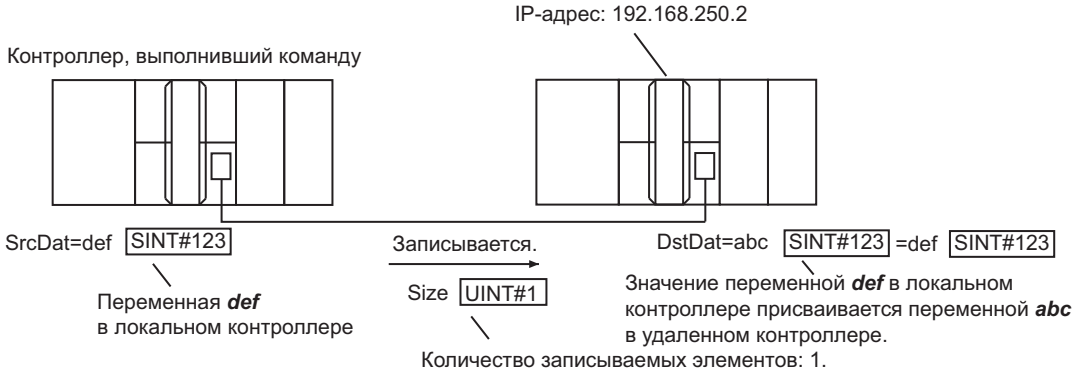
LD



ST

```
CIPUCMMWrite_instance(A, '2\192.168.250.2',
                      UINT#0, 'abc', UINT#1,
                      def, ghi, jkl, mno, pqr,
                      stu);
```

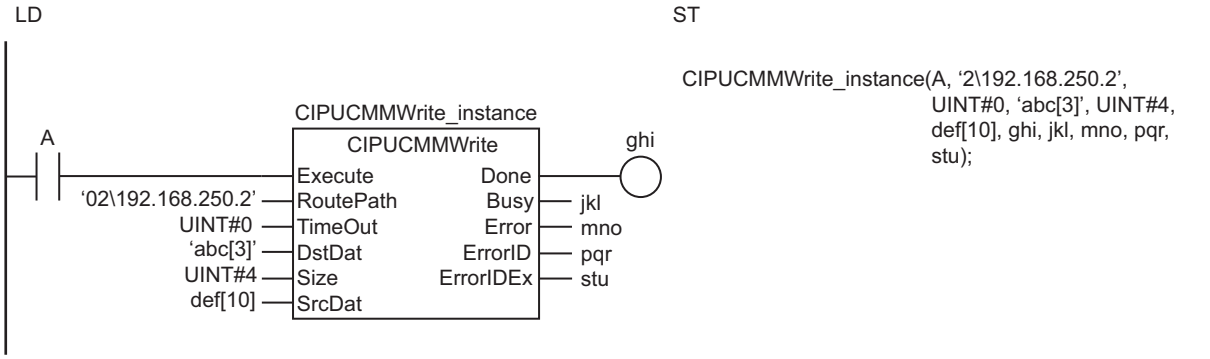
Значение переменной ***SrcDat*** в локальном контроллере присваивается переменной ***DstDat*** в удаленном контроллере в сети CIP, указанном параметром ***RoutePath*** (путь маршрута).
Количество записываемых элементов указывается параметром ***Size***.



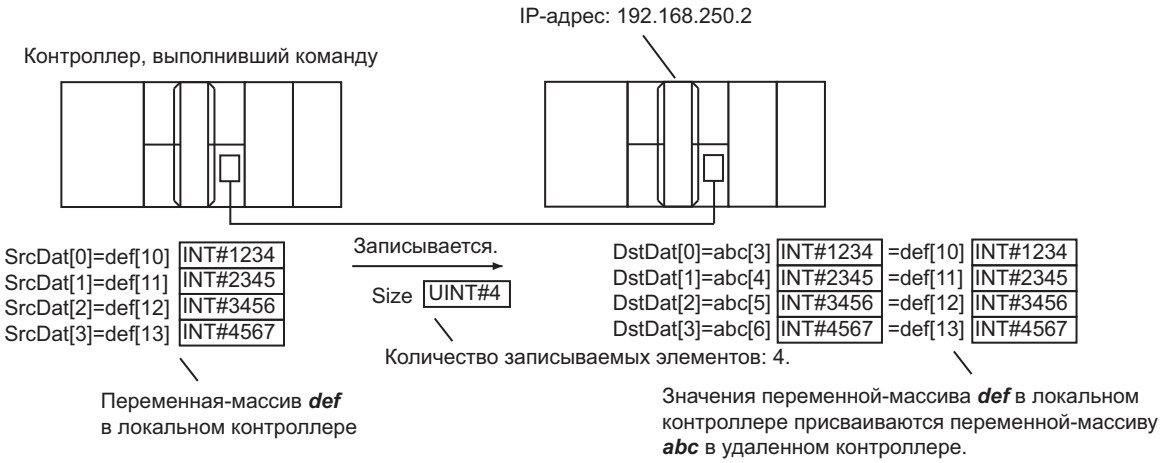
Запись массивов

Для записи данных массива в переменную *DstDat* в качестве параметра следует передать элемент массива с указанием индекса. В переменную *SrcDat* также следует передать в качестве параметра элемент массива с указанием индекса.

В следующем примере содержимое элементов переменной-массива def[10]...def [13] сохраняется в четыре элемента переменной-массива abc[3]...abc[6].



Значения элементов переменной-массива ***def[10]…def[13]*** в локальном контроллере присваиваются элементам переменной-массива ***abc[3]…abc[6]*** в удаленном контроллере.



Максимальный объем записываемых данных

Максимальный объем данных, которые могут быть записаны, зависит от типа данных, а также от имени переменной, которое указано в *DstDat*, и от пути маршрута. Сказанное отражено в следующей таблице.

Максимальный объем записываемых данных [байт] = базовый объем - размер имени переменной в *DstDat* - размер информации о пути

Параметр в формуле выше	Значение
Базовый объем	• Тип данных переменной, указанной для *DstDat* — структура: 492 байт • Тип данных переменной, указанной для *DstDat* — STRING: 494 байт • Другие типы данных: 496 байт

Параметр в формуле выше	Значение
Размер имени переменной в *DstDat*	• Размер имени переменной рассчитывается как общее количество байтов для символов ASCII на всех уровнях структуры плюс удвоенное количество уровней. • Если количество байтов символов ASCII на некотором уровне нечетное, следует добавить 1. • Если некоторый уровень в структуре является массивом, следует добавить количество измерений в массиве, помноженное на 4. • Точки и запятые в структуре и массивах в размер имени переменной не включаются. Пример 1. Имя переменной в *DstDat*: «aaa.bbbbb[1,2,3].cc» • Размер текстовой строки "aaa" на первом уровне составляет 3 байта. Это нечетное число, поэтому добавляем к нему 1 и получаем 4 байта. • "bbbbb" в "bbbbb[1,2,3]" на втором уровне — это текстовая строка размером в 5 байтов. Это нечетное число, поэтому добавляем к нему 1 и получаем 6 байтов. • Кроме того, "bbbbb[1,2,3]" — это трехмерный массив, поэтому также добавляем 12 (3 x 4) и получаем в итоге 18 байтов. • Размер текстовой строки "cc" на третьем уровне составляет 2 байта. Это четное число, поэтому при вычислении используется 2 байта. • Умножим количество уровней 3 на 2 и получим 6. Теперь сложим все полученные значения: 6 + 4 байта для первого уровня + 18 байтов для второго уровня + 2 байта для третьего уровня. Итого, размер имени переменной составляет 30 байтов. Пример 2. Имя переменной в *DstDat*: «val» • Размер текстовой строки "val" на первом уровне составляет 3 байта. Это нечетное число, поэтому добавляем к нему 1 и получаем 4 байта. • Умножим количество уровней 1 на 2 и получим 2. Следовательно, размер имени переменной составляет 2 + 4 = 6 байтов. Пример 3. Имя переменной в *DstDat*: «array[8]» • Размер текстовой строки "array" на первом уровне составляет 5 байтов. Это нечетное число, поэтому добавляем к нему 1 и получаем 6 байтов. • Это одномерный массив. Поэтому также нужно добавить 4 (результат умножения 1 на 4). • Умножим количество уровней 1 на 2 и получим 2. Следовательно, размер имени переменной составляет 2 + 6 + 4 = 12 байтов.

Параметр в формуле выше	Значение
Размер информации о пути	• При отсутствии транзитных участков размер информации о пути равен 0 байтов. *1 • Если имеются транзитные участки, размер информации о пути равен размеру пути маршрута, к которому нужно добавить 12 байтов. • Размер пути маршрута определяется как количество байтов символов ASCII в пути маршрута. • Однако действуют следующие дополнительные правила: a) Если адресная часть начинается с символа "#", общий размер сетевой и адресной частей следует принять равным 2 байтам. b) Если адресная часть не начинается с символа "#", размер сетевой части нужно принять равным 2 байтам. c) Если адресная часть не начинается с символа "#", а количество байтов в символах ASCII адресной части является нечетным числом, следует добавить 1 байт. d) В размер пути маршрута не следует включать знак "\", разделяющий уровни в пути маршрута. e) В размер пути маршрута не следует включать первый транзитный участок. Пример 1. Путь маршрута: «01\#11\02\192.168.250.2\01\#01» • Первый транзитный участок при определении размера пути маршрута не учитывается, поэтому часть "01\#11" в начале пути в расчет не принимаем. • Указан тип сети "02", поэтому для расчета используем 2 байта. • Размер адресной части "192.168.250.2" составляет 13 байтов. Это нечетное число, поэтому добавляем к нему 1 и получаем 14 байтов. • В следующем фрагменте ("01\#01") адресная часть начинается с символа "#", поэтому общий размер сетевой и адресной частей для расчета принимается равным 2 байтам. • Сложим все полученные размеры: 2 + 14 + 2. Итого, размер пути маршрута составляет 18 байтов. • Добавив к размеру пути маршрута 12 байтов, получим размер информации о пути: 18 + 12 = 30 байтов. Пример 2. Путь маршрута: «02\192.168.250.2\01\#00» • Первый транзитный участок при определении размера пути маршрута не учитывается, поэтому часть "02\192.168.250.2" в начале пути в расчет не принимаем. • В следующем фрагменте ("01\#00") адресная часть начинается с символа "#", поэтому общий размер сетевой и адресной частей для расчета принимается равным 2 байтам. • Следовательно, размер пути маршрута составляет 2 байта. • Добавив к размеру пути маршрута 12 байтов, получим размер информации о пути: 2 + 12 = 14 байтов. Пример 3. Путь маршрута: «02\192.168.250.2» • При отсутствии транзитных участков размер информации о пути равен 0 байтов.

*1. Транзитный участок (также «переход» или «хоп») — это минимальная единица маршрута, единичный участок маршрута между двумя узлами: передающим узлом и принимающим узлом. Например, если указан путь маршрута «02\192.168.250.2\01\#00», сообщение сначала направляется на узел с IP-адресом 192.168.250.2, а затем передается модулю с адресом 00. Этот маршрут включает один транзитный участок.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.

*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения об интерфейсе связи CIP см. в следующих документах.

- *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*
- *Серия CJ, модули EtherNet/IP — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W495)*

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Всегда используйте переменную в качестве входного параметра, передаваемого в *SrcDat*. При передаче константы произойдет ошибка сборки.
- Если параметр *SrcDat* является перечислением, в него нельзя напрямую передавать перечислитель. В случае непосредственной передачи перечислителя произойдет ошибка сборки.
- Эту команду можно использовать для встроенного порта EtherNet/IP модуля ЦПУ серии NJ/NX или порта модуля интерфейса EtherNet/IP, подключенного к модулю ЦПУ серии NJ.
- Если переменная записывается в контроллер OMRON, она должна быть открыта для доступа по сети («опубликована»). Это должно быть сделано заранее.
- Для записи данных нельзя указать непосредственно значение адреса памяти, используемой для модулей серии CJ. Если нужно записать данные по конкретному адресу в памяти, используемой для модулей серии CJ, необходимо заранее назначить переменной требуемый адрес в памяти с помощью параметра «AT».
- Нельзя указать непосредственно значение адреса локальной памяти, используемой для модулей серии CJ. Если нужно получить данные по конкретному адресу в памяти, используемой для модулей серии CJ, необходимо заранее назначить переменной *SrcDat* требуемый адрес в памяти с помощью параметра «AT».
- В следующей таблице указаны символы, которые можно использовать в *DstDat*.

Параметр	Требования
Максимальное количество байтов	127 байт
Кодировка	UTF-8
Применимые символы	Буквенно-цифровые символы (регистр не учитывается), однобайтовые символы катаканы, многобайтовые символы и «_» (символы подчеркивания)

Параметр	Требования
Недопустимые текстовые строки	• Любая текстовая строка, начинающаяся с символа от 0 до 9 (коды символов от 16#30 до 16#39) ASCII. • Текстовая строка, состоящая только из одного символа «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, содержащая подряд два и более символов «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, начинающаяся с символа «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, заканчивающаяся символом «_» (нижнее подчеркивание) ASCII. • Любая текстовая строка, начинающаяся с «P_».

- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение *TimeOut* находится за пределами допустимого диапазона.
  b) Значение *Size* находится за пределами допустимого диапазона.
  c) В параметр *DstDat* передана недопустимая текстовая строка.
  d) Значение *Size* превышает размер переменной *SrcDat*.
  e) Для переменной *SrcDat* был указан неподдерживаемый тип данных.
  f) Был возвращен ответ с ошибкой, которая определена в протоколе CIP.
  g) В параметр *RoutePath* передана недопустимая текстовая строка.
  h) Одновременно было выполнено более 32 команд, связанных с протоколом CIP.
  i) Ответ не был получен в течение заданного времени ожидания.
  j) Имеется ошибка в настройке локального IP-адреса.
  k) Произошла ошибка дублирования IP-адреса.
- Для этой команды в качестве дополнительного кода ошибки в переменной *ErrorIDEx* указывается код ошибки сообщения CIP. Значения кодов ошибки приведены в таблице ниже.

Значение	Ошибка
16#02000000	Нормальный обмен данными невозможен из-за высокой нагрузки на удаленном узле.
16#04000000	Для чтения указана переменная одного из следующих типов данных, который не предусмотрен в другом контроллере. • Базовый тип данных • Перечисление • Структура • Объединение • Массив
16#05000000	Для чтения указана переменная одного из следующих типов, который не предусмотрен в другом контроллере. • Перечислитель перечисления • Член структуры • Член объединения • Элемент массива
16#08000000	Запрошенная служба не поддерживается.
16#0C008010	Указанная адресуемая переменная в данный момент загружается.
16#0C008011	
16#1F000102	• Указанная адресуемая переменная имеет атрибут Constant (Константа), поэтому запись в нее невозможна. • Записываемые данные не соответствуют количеству записываемых элементов.
16#1F008007	Указанная переменная недоступна.
16#20008017	Указанная адресуемая переменная не является массивом, при этом заданное количество элементов для записи не равно 1.
16#20008018	Указанная адресуемая переменная является массивом, при этом заданное количество элементов для записи превышает фактическое количество элементов в массиве.

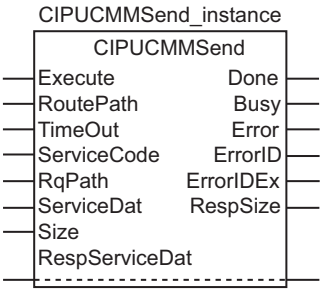
Значение	Ошибка
16#20008028	• Указанная адресуемая переменная является перечислением, при этом записываемое значение не является значением перечислителя. •
16#26000000	Указанное имя адресуемой переменной содержит только пустой символ (NULL).

Пример программы

См. *Пример программы* на стр. 2-1220 для команды CIPUCMMSend.

CIPUCMMSend

Команда CIPUCMMSend передает сообщение UCMM по протоколу CIP указанному устройству в сети CIP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
CIPUCMMSend	Передача явного сообщения UCMM	FB		CIPUCMMSend_instance(Execute , RoutePath, TimeOut, ServiceCode, RqPath, ServiceDat, Size, RespServiceDat, Done, Busy, Error, ErrorID, ErrorIDEx, RespSize);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
RoutePath	Путь маршрута	Вход	Путь маршрута	Зависит от типа данных.	---	---
TimeOut	Время ожидания		Время ожидания	1…65535	0,1 с	20 (2,0 с)
ServiceCode	Код службы		Код службы	Зависит от типа данных.	---	---
RqPath	Путь запроса		Путь запроса	---		
ServiceDat	Данные команды		Передаваемые данные	Зависит от типа данных.		*1
Size	Количество элементов для передачи		Количество элементов для передачи			1
RespServiceDat	Данные ответа	Вход-выход	Данные ответа	Зависит от типа данных.	---	---
RespSize	Размер ответа	Выход	Объем данных ответа	Зависит от типа данных.	Байты	---

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RoutePath																				ОК
TimeOut							ОК													

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ServiceCode		ОК																		
RqPath	Структура _sREQUEST_PATH или _sREQUEST_PATH_EX*1. Дополнительные сведения см. в разделе *Тип данных переменной RqPath* на стр. 2-1193.																			
ServiceDat		ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
	Также можно указать массив, член структуры или член объединения.																			
Size							ОК													
RespServiceDat		ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК					
	Также можно указать массив, член структуры или член объединения.																			
RespSize							ОК													

*1. Чтобы можно было указать тип _sREQUEST_PATH_EX, требуется модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Функция

Команда CIPUCMMSend передает данные команды *ServiceDat* для службы, указанной кодом службы *ServiceCode*, в виде явного сообщения класса UCMM.
Адресат данных указывается с помощью параметра *RoutePath* (путь маршрута).
В параметре *RqPath* указывается путь запроса.

В параметре *Size* указывается количество элементов, которые должны быть переданы.
Если *ServiceDat* является массивом, следует указать количество элементов массива, которые должны быть переданы.
Если же *ServiceDat* не является массивом, всегда следует указывать значение *1*.
Если передавать данные службы не требуется, *Size* можно задать равным *0*.

Принятые в ответ данные сохраняются в переменную *RespServiceDat*. Количество байтов данных ответа записывается в переменную *RespSize*.

Параметр *TimeOut* задает время ожидания. Если ответ не возвращается в течение времени ожидания, считается, что произошел сбой связи.

Для параметров *ClassIDLogicalFormat*, *InstanceIDLogicalFormat* и *AttributeIDLogicalFormat* используется перечислимый тип данных _eCIP_LOGICAL_FORMAT.
Значения перечислителей перечислимого типа _eCIP_LOGICAL_FORMAT приведены в таблице ниже:

Перечислитель	Значение
_8BIT	8 бит
_16BIT	16 бит
_32BIT	32 бит

Если значение *ErrorID* = WORD#16#1C00, в переменную *ErrorIDEx* сохраняется код ошибки сообщения CIP.
Значение кода ошибки в *ErrorIDEx* и его смысл зависят от конкретного удаленного узла. См. руководство по соответствующему удаленному узлу.

Тип данных переменной *RqPath*

Для переменной *RqPath* используется структурный тип данных _sREQUEST_PATH или _sREQUEST_PATH_EX.
Как правило, используется тип _sREQUEST_PATH.
При указании какого-либо размера логического формата следует использовать тип _sREQUEST_PATH_EX.

● Тип данных _sREQUEST_PATH

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
RqPath	Путь запроса	Путь запроса	_sREQUEST_PATH	---	---	---
ClassID	Идентификатор класса	Идентификатор класса	UINT	Зависит от типа данных.	---	0
InstanceID	Идентификатор экземпляра	Идентификатор экземпляра	UINT			0
isAttributeID	Использование атрибута	ИСТИНА: используется идентификатор атрибута. ЛОЖЬ: идентификатор атрибута не используется.	BOOL			ЛОЖЬ
AttributeID	Идентификатор атрибута	Идентификатор атрибута	UINT			0

Примечание Размер логического формата каждого идентификатора в типе _sREQUEST_PATH составляет 16 бит.

● **Тип данных _sREQUEST_PATH_EX**

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
RqPath	Путь запроса	Путь запроса	_sREQUEST_PATH_EX	---	---	---
ClassIDLogicalFormat	Логический формат идентификатора класса	Размер данных идентификатора класса	_eCIP_LOGICAL_FORMAT	Зависит от типа данных.	---	_8BIT
ClassID	Идентификатор класса	Идентификатор класса	UDINT			0
InstanceIDLogicalFormat	Логический формат идентификатора экземпляра	Размер данных идентификатора экземпляра	_eCIP_LOGICAL_FORMAT			_8BIT
InstanceID	Идентификатор экземпляра	Идентификатор экземпляра	UDINT			0
isAttributeID	Использование атрибута	ИСТИНА: используется идентификатор атрибута. ЛОЖЬ: идентификатор атрибута не используется.	BOOL			ЛОЖЬ
AttributeIDLogicalFormat	Логический формат идентификатора атрибута	Размер данных идентификатора атрибута	_eCIP_LOGICAL_FORMAT			_8BIT
AttributeID	Идентификатор атрибута	Идентификатор атрибута	UDINT			0

Передача и прием массивов

Если переменная *ServiceDat* или *RespServiceDat* является массивом, в нее в качестве параметра следует передать элемент массива с указанием индекса.

Максимальный объем читаемых/записываемых данных

Может быть прочитано максимум 492 байта данных.
Максимальный объем данных, которые могут быть записаны, зависит от того, имеется ли атрибут пути запроса, а также от используемого пути запроса, что отражено в таблице ниже.

Максимальный объем записываемых данных [байт] = базовый объем - объем, зависящий от использования атрибута - размер информации о пути

Параметр в формуле выше	Значение
Базовый объем	500 байт
Использование атрибута*1	Идентификатор атрибута используется: 14 байт Идентификатор атрибута не используется: 10 байт
Размер информации о пути	• При отсутствии транзитных участков размер информации о пути равен 0 байтов. *2 • Если имеются транзитные участки, размер информации о пути равен размеру пути маршрута, к которому нужно добавить 12 байтов. • Размер пути маршрута определяется как количество байтов символов ASCII в пути маршрута. • Однако действуют следующие дополнительные правила: a) Если адресная часть начинается с символа "#", общий размер сетевой и адресной частей следует принять равным 2 байтам. b) Если адресная часть не начинается с символа "#", размер сетевой части нужно принять равным 2 байтам. c) Если адресная часть не начинается с символа "#", а количество байтов в символах ASCII адресной части является нечетным числом, следует добавить 1 байт. d) В размер пути маршрута не следует включать знак "\", разделяющий уровни в пути маршрута. e) В размер пути маршрута не следует включать первый транзитный участок. Пример 1. Путь маршрута: «01\#11\02\192.168.250.2\01\#01» • Первый транзитный участок при определении размера пути маршрута не учитывается, поэтому часть "01\#11" в начале пути в расчет не принимаем. • Указан тип сети "02", поэтому для расчета используем 2 байта. • Размер адресной части "192.168.250.2" составляет 13 байтов. Это нечетное число, поэтому добавляем к нему 1 и получаем 14 байтов. • В следующем фрагменте ("01\#01") адресная часть начинается с символа "#", поэтому общий размер сетевой и адресной частей для расчета принимается равным 2 байтам. • Сложим все полученные размеры: 2 + 14 + 2. Итого, размер пути маршрута составляет 18 байтов. • Добавив к размеру пути маршрута 12 байтов, получим размер информации о пути: 18 + 12 = 30 байтов. Пример 2. Путь маршрута: «02\192.168.250.2\01\#00» • Первый транзитный участок при определении размера пути маршрута не учитывается, поэтому часть "02\192.168.250.2" в начале пути в расчет не принимаем. • В следующем фрагменте ("01\#00") адресная часть начинается с символа "#", поэтому общий размер сетевой и адресной частей для расчета принимается равным 2 байтам. • Следовательно, размер пути маршрута составляет 2 байта. • Добавив к размеру пути маршрута 12 байтов, получим размер информации о пути: 2 + 12 = 14 байтов. Пример 3. Путь маршрута: «02\192.168.250.2» • При отсутствии транзитных участков размер информации о пути равен 0 байтов.

*1. В случае модуля ЦПУ с версией модуля 1.10 или более ранней или Sysmac Studio версии 1.14 или ниже применяются следующие значения:
Идентификатор атрибута используется: 12 байт
Идентификатор атрибута не используется: 8 байт

*2. Транзитный участок (также «переход» или «хоп») — это минимальная единица маршрута, единичный участок маршрута между двумя узлами: передающим узлом и принимающим узлом. Например, если указан путь маршрута «02\192.168.250.2\01\#00», сообщение сначала направляется на узел с IP-адресом 192.168.250.2, а затем передается модулю с адресом 00. Этот маршрут включает один транзитный участок.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения об интерфейсе связи CIP см. в следующих документах.

- *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*
- *Серия CJ, модули EtherNet/IP — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W495)*

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения выполнения команды.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Всегда используйте переменную в качестве входного параметра, передаваемого в *ServiceDat*. При передаче константы произойдет ошибка сборки.
- Эту команду можно использовать для встроенного порта EtherNet/IP модуля ЦПУ серии NJ/NX или порта модуля интерфейса EtherNet/IP, подключенного к модулю ЦПУ серии NJ.
- Если переменная записывается в контроллер OMRON, она должна быть открыта для доступа по сети («опубликована»). Это должно быть сделано заранее.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для параметра *RqPath.ClassIDLogicalFormat* или *RqPath.AttributeIDLogicalFormat* указано значение, выходящее за пределы допустимого диапазона.
  b) Возникло несоответствие между следующими двумя переменными: размер, указанный в *RqPath.ClassIDLogicalFormat*, и размер значения *RqPath.ClassID*; размер, указанный в *RqPath.InstanceIDLogicalFormat*, и размер значения *RqPath.InstanceID*; или размер, указанный в *RqPath.AttributeIDLogicalFormat*, и размер значения *RqPath.AttributeID*.
  c) Значение *TimeOut* находится за пределами допустимого диапазона.
  d) Значение *Size* превышает размер записываемых данных.
  e) Значение *Size* превышает размер переменной *ServiceDat*.
  f) Значение *RespSize* превышает размер переменной *RespServiceDat*.
  g) Для переменной *ServiceDat* был указан неподдерживаемый тип данных.
  h) Для переменной *RespServiceDat* был указан неподдерживаемый тип данных.
  i) Тип данных переменной, указанной для пути запроса *RqPath*, отличается от _sREQUEST_PATH и _sREQUEST_PATH_EX.

j) Имеется ошибка в настройке локального IP-адреса.
k) Произошла ошибка дублирования IP-адреса.
l) Команда была выполнена при ошибке сервера BOOTP.
m) Был возвращен ответ с ошибкой, которая определена в протоколе CIP.
n) В параметр *RoutePath* передана недопустимая текстовая строка.
o) Одновременно было выполнено более 32 команд, связанных с протоколом CIP.
p) Ответ не был получен в течение заданного времени ожидания.

Пример программы

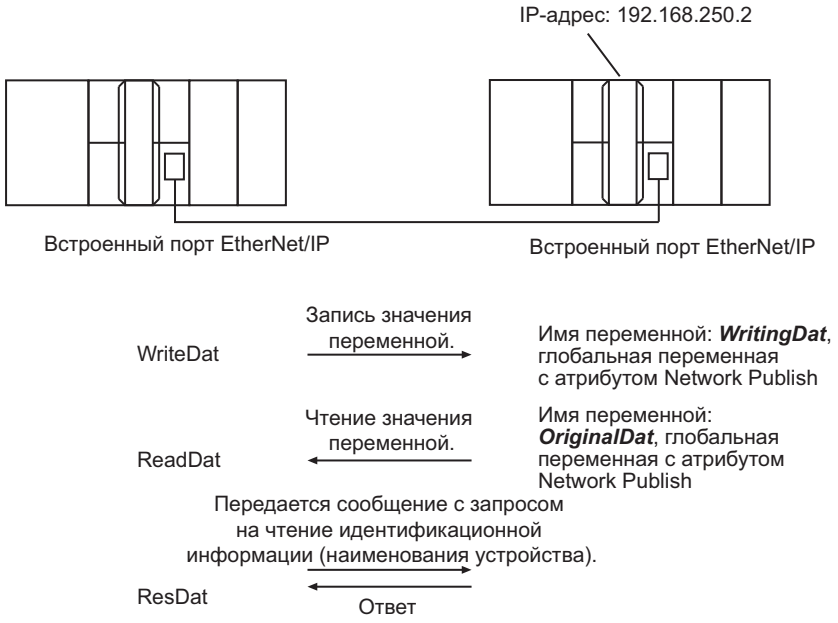
Далее будет рассмотрен пример применения сообщений UCMM протокола CIP для записи значения в переменную, чтения значения переменной и передачи сообщения.
Контроллеры подключены к сети EtherNet/IP. IP-адрес удаленного узла: 192.168.250.2.
Соблюдается приведенный ниже порядок действий.

1 Используется команда CIPUCMMWrite для записи значения в переменную на удаленном узле. Переменная на удаленном узле носит имя *WritingDat*, в нее записывается содержимое переменной *WriteDat*.
Переменная *WritingDat* должна быть определена на удаленном узле как глобальная переменная, и для нее должен быть настроен атрибут Network Publish (Публикация в сети).

2 Используется команда CIPUCMMRead для чтения значения переменной на удаленном узле. Производится чтение значения переменной *OriginalDat* на другом узле, и прочитанное значение сохраняется в переменную *ReadDat*.
Переменная *OriginalDat* должна быть определена на удаленном узле как глобальная переменная, и для нее должен быть настроен атрибут Network Publish (Публикация в сети).

3 Используется команда CIPUCMMSend для передачи явного сообщения удаленному узлу. Сообщение содержит запрос на чтение идентификационной информации (наименования устройства).
Значения идентификатора класса, идентификатора экземпляра, идентификатора атрибута и кода службы приведены в таблице ниже. Возвращенные данные сохраняются в переменную *ResDat*.

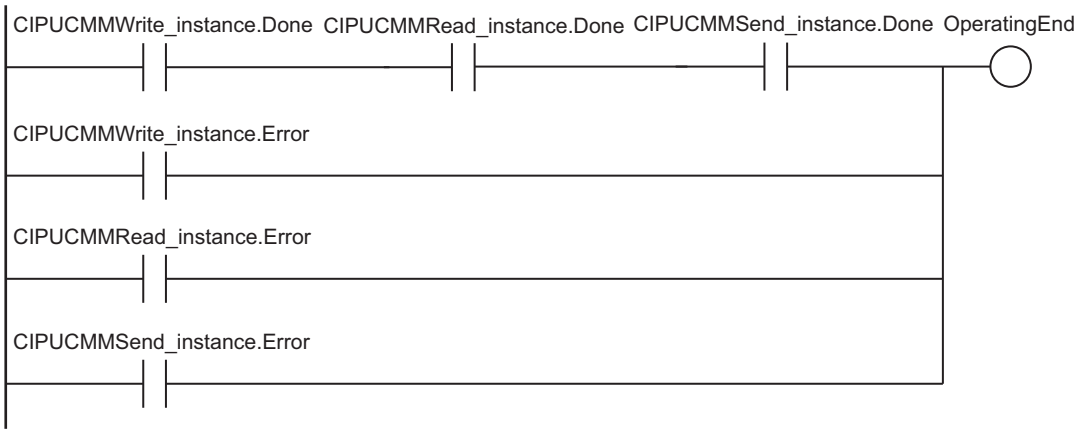
Параметр	Значение
Идентификатор класса	1
Идентификатор экземпляра	1
Идентификатор атрибута	7
Код службы	16#0E



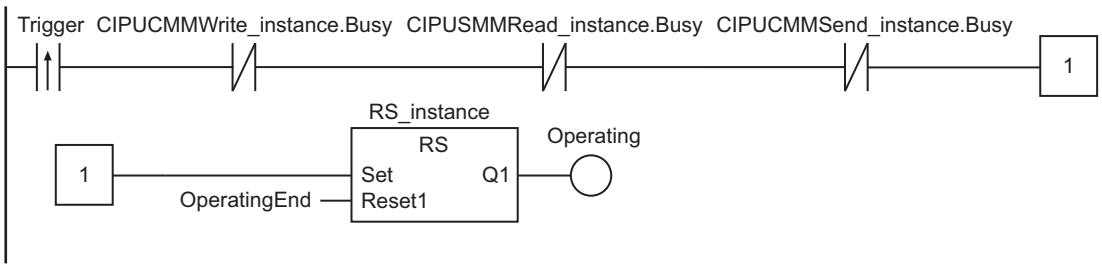
Программа на языке LD

Переменная	Тип данных	Начальное значение	Комментарий
OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена
Trigger	BOOL	ЛОЖЬ	Условие выполнения
Operating	BOOL	ЛОЖЬ	Обработка
WriteDat	INT	1234	Записываемые данные
ReadDat	INT	0	Прочитанные данные
ReqPath	_sREQUEST_PATH	(ClassID:=0, InstanceID:=0, isAttributeID:=FALSE, AttributeID:=0)	Путь запроса
ResDat	ARRAY[0..10] OF BYTE	[11(16#0)]	Данные ответа
Формальная переменная	BYTE	16#0	Формальная переменная
RS_instance	RS		
CIPUCMMWrite_instance	CIPUCMMWrite		
CIPUCMMRead_instance	CIPUCMMRead		
CIPUCMMSend_instance	CIPUCMMSend		

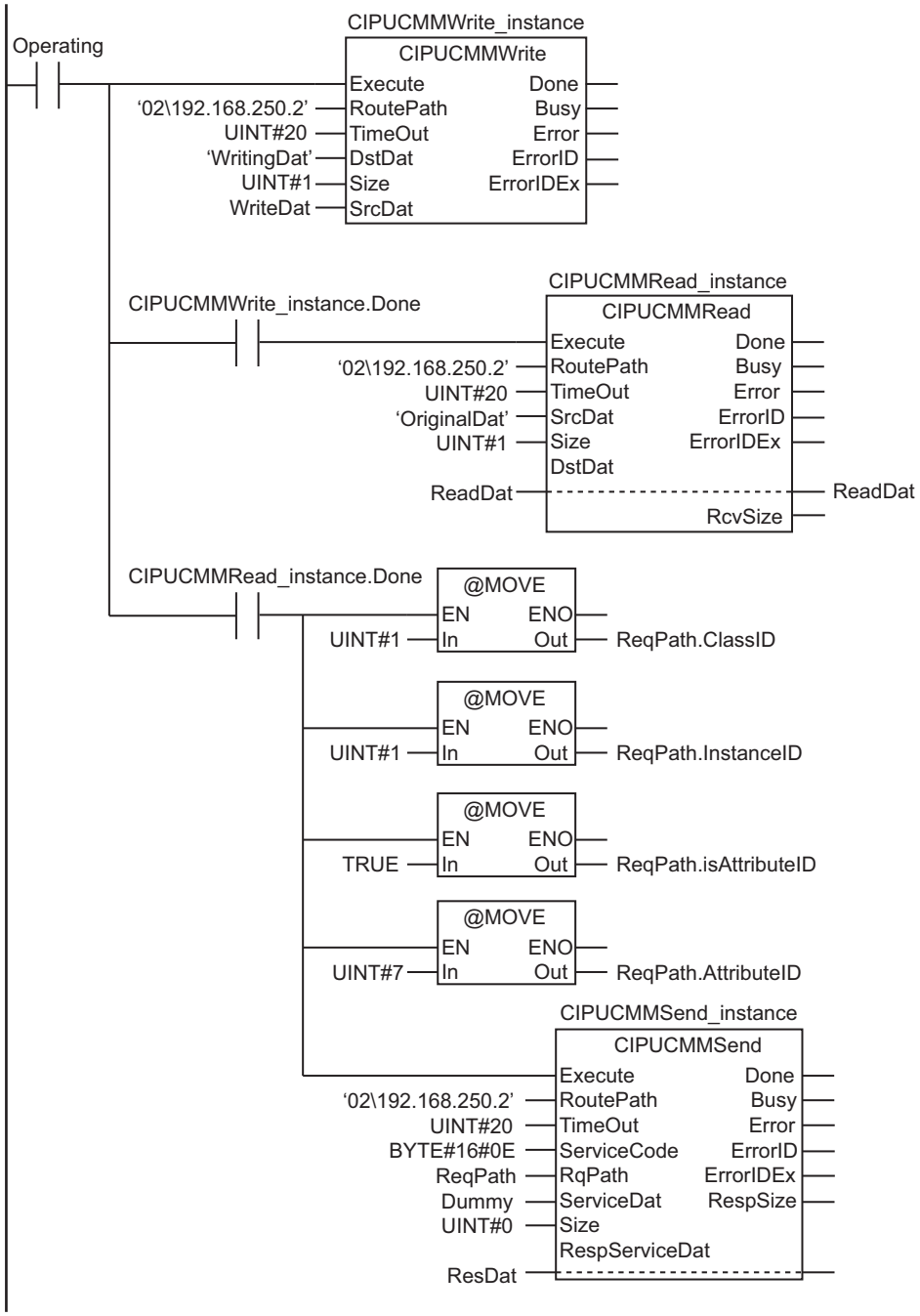
Определение, завершено ли выполнение команды.



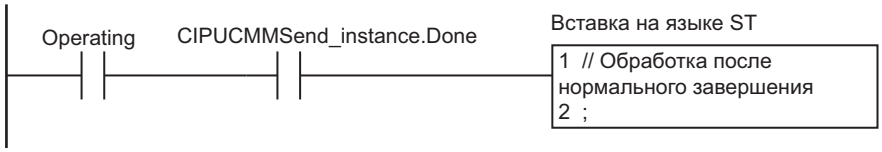
Прием условия выполнения.

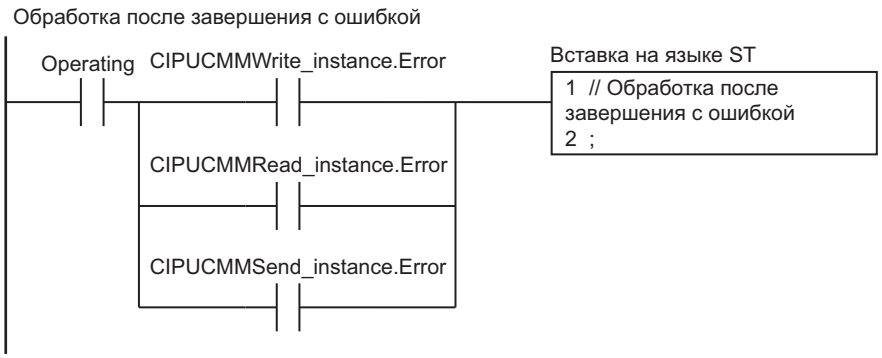


Выполнение команды



Обработка после нормального завершения





Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	DoUCMMTrigger	BOOL	ЛОЖЬ	Обработка
	Stage	INT	0	Смена этапа
	WriteDat	INT	0	Записываемые данные
	ReadDat	INT	0	Прочитанные данные
	ReqPath	_sREQUEST_PATH	(ClassID:=0, InstanceID:=0, isAttributeID:=FALSE, AttributeID:=0)	Путь запроса
	ResDat	ARRAY[0..10] OF BYTE	[11(16#0)]	Данные ответа
	Формальная переменная	BYTE	16#0	Формальная переменная
	CIPUCMMWrite_instance	CIPUCMMWrite		
	CIPUCMMRead_instance	CIPUCMMRead		
	CIPUCMMSend_instance	CIPUCMMSend		

Внешние переменные	Переменная	Константа	Тип данных	Комментарий
	_EIP_EtnOnlineSta	☑	BOOL	В сети

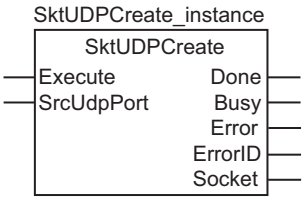
```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (DoUCMMTrigger=FALSE) AND (_Eip_EtnOnlineSta=TRUE) ) THEN
    DoUCMMTrigger         :=TRUE;
    Stage                 :=INT#1;
    CIPUCMMWrite_instance(
        Execute           :=FALSE,                  // Инициализация экземпляра.
        SrcDat            :=WriteDat);              // Формальная запись
    CIPUCMMRead_instance(                           // Инициализация экземпляра.
```

```
        Execute          :=FALSE,                  // Формальная запись
        DstDat           :=ReadDat);               // Формальная запись
    CIPUCMMSend_instance(
        Execute          :=FALSE,                  // Инициализация экземпляра.
        ServiceDat       := Dummy,                 // Формальная запись
        RespServiceDat:=ResDat);                   // Формальная запись
END_IF;

IF (DoUCMMTrigger=TRUE) THEN
    CASE Stage OF
    1 :                                    // Запрос записи значения переменной.
        CIPUCMMWrite_instance(
            Execute  :=TRUE,
            RoutePath:='02\192.168.250.2',     // Путь маршрута
            TimeOut  :=UINT#20,                // Время ожидания
            DstDat   :='WritingDat',           // Имя переменной назначения
            Size     :=UINT#1,                 // Количество элементов для записи
            SrcDat   :=WriteDat);              // Записываемые данные

        IF (CIPUCMMWrite_instance.Done=TRUE) THEN
            Stage:=INT#2;                              // Нормальное завершение
        ELSIF (CIPUCMMWrite_instance.Error=TRUE) THEN
            Stage:=INT#10;                             // Завершение с ошибкой
        END_IF;
    2 :                                    // Запрос чтения значения переменной.
        CIPUCMMRead_instance(
            Execute   :=TRUE,
            RoutePath :='02\192.168.250.2',            // Путь маршрута
            TimeOut   :=UINT#20,                       // Время ожидания
            SrcDat    :='OriginalDat',                 // Имя адресуемой переменной
            Size      :=UINT#1,                        // Количество элементов для ч
тения
            DstDat    :=ReadDat);                      // Прочитанные данные

        IF (CIPUCMMRead_instance.Done=TRUE) THEN
            Stage:=INT#3;                            // Нормальное завершение
        ELSIF (CIPUCMMRead_instance.Error=TRUE) THEN
            Stage:=INT#40;                           // Завершение с ошибкой
        END_IF;

    3 :                                    // Передача сообщения
        ReqPath.ClassID      :=UINT#01;
        ReqPath.InstanceID   :=UINT#01;
        ReqPath.isAttributeID:=TRUE;
        ReqPath.AttributeID  :=UINT#07;
        CIPUCMMSend_instance(
            Execute        :=TRUE,
            RoutePath      :='02\192.168.250.2',       // Путь маршрута
            TimeOut        :=UINT#20,                  // Время ожидания
            ServiceCode    :=BYTE#16#0E,               // Код службы
            RqPath         :=ReqPath,                  // Путь запроса
```

```
            ServiceDat     := Dummy,                    // Данные службы
            Size           :=UINT#0,                    // Количество элементов
            RespServiceDat :=ResDat);                   // Ответные данные

        IF (CIPUCMMSend_instance.Done=TRUE) THEN
            Stage:=INT#0;                              // Нормальное завершение
        ELSIF (CIPUCMMSend_instance.Error=TRUE) THEN
            Stage:=INT#30;                             // Завершение с ошибкой
        END_IF;

    0 :                                  // Обработка после нормального завершения
        DoUCMMTrigger:=FALSE;
        Trigger      :=FALSE;

    ELSE                                          // Обработка после завершения с ош
ибкой
        DoUCMMTrigger:=FALSE;
        Trigger      :=FALSE;
    END_CASE;
END_IF;
```

SktUDPCreate

Команда SktUDPCreate создает запрос на открытие сокета UDP для открытия порта UDP на встроенном порте EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktUDPCreate	Создание сокета UDP	FB		SktUDPCreate_instance(Execute, SrcUdpPort, Done, Busy, Error, ErrorID, Socket);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
SrcUdpPort	Номер локального порта UDP	Вход	Номер локального порта UDP	1…65535	---	1
Socket	Сокет	Выход	Сокет	---	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SrcUdpPort							ОК													
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1227.																			

Функция

Команда SktUDPCreate открывает порт, указанный параметром *ScrUdpPort* (номер локального порта UDP). Для этого она выполняет функции сокетов Socket() и Bind().
Информация об открытом сокете сохраняется в выходную переменную *Socket*.
Если выполнение команды завершается нормально, значение переменной *Done* меняется на ИСТИНА.
Порт UDP открывается, если команда завершается нормально.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор для обмена данными	UDINT	Зависит от типа данных.	---	0
SrcAdr*1	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.	---	''
DstAdr*1	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.	---	''

*1. В эти члены структуры выводится значение *0* или NULL.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Для закрытия дескрипторов, созданных с помощью этой команды, используйте команду SktClose.
- При переходе в режим «Программирование» созданные этой командой дескрипторы деактивируются.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Имеется ошибка в настройке локального IP-адреса.
  b) Значение *SrcUdpPort* находится за пределами допустимого диапазона.
  c) Порт, указанный параметром *SrcUdpPort*, уже открыт или для него в настоящее время выполняется операция закрытия.
  d) Порт, указанный параметром *SrcUdpPort*, уже используется.

Информация о версии

- Количество сокетов, которые могут быть открыты одновременно, зависит от версии модуля ЦПУ, как показано в следующей таблице. Это общие предельные значения в сумме для сокетов UDP и TCP.

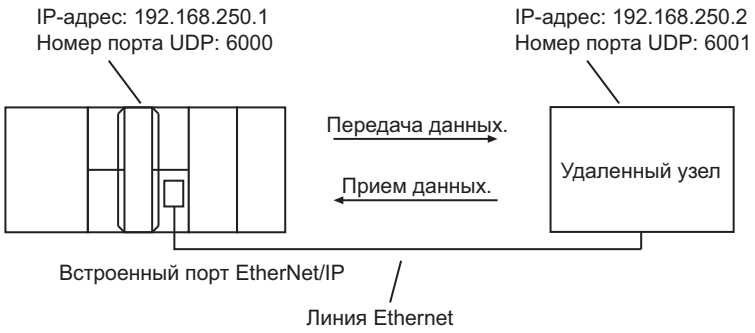
Версия модуля ЦПУ	Количество сокетов
1.03 или более поздняя версия	Макс. 30*1
1.02 или более ранняя версия	Макс. 16

*1. В случае модулей ЦПУ NX102 может быть открыто максимум 60 сокетов.

- При использовании модуля ЦПУ версии 1.10 или более поздней версии значение *Socket* не изменяется, даже если значение *Error* меняется на ИСТИНА. В случае версии 1.09 или более ранней версии значение *Socket* меняется на *0*.

Пример программы

Рассмотрим пример программы для случая, когда для обмена данными между Серия NJ/NX, модули ЦПУ и удаленным узлом используется служба сокета UDP.



Программа пользователя Серия NJ/NX, модули ЦПУ

Соблюдается следующий порядок действий:

1 С помощью команды SktUDPCreate запрашивается создание сокета UDP.

2 С помощью команды SktUDPSend запрашивается передача данных. Передаются данные, содержащиеся в массиве SendSocketDat[].

3 С помощью команды SktUDPRcv запрашивается прием данных. Принятые данные сохраняются в массив RcvSocketDat[].

4 Используется команда SktClose для закрытия сокета.

● Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	DoSendAndRcv	BOOL	ЛОЖЬ	Обработка
	Stage	INT	0	Смена этапа
	RcvSocketDat	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Принимаемые данные
	WkSocket	_sSOCKET	(Handle:=0, SrcAdr:=(PortNo:=0, IpAdr:=''), DstAdr:=(PortNo:=0, IpAdr:=''))	Сокет
	SendSocketDat	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Передаваемые данные
	SktUDPCreate_instance	SktUDPCreate		
	SktUDPSend_instance	SktUDPSend		
	SktUDPRcv_instance	SktUDPRcv		
	SktClose_instance	SktClose		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EIP_EtnOnlineSta	BOOL	☑	В сети

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (DoSendAndRcv=FALSE) AND (_Eip_EtnOnlineSta=TRUE) ) THEN
    DoSendAndRcv:=TRUE;
    Stage       :=INT#1;
    SktUDPCreate_instance(Execute:=FALSE);    // Инициализация экземпляра.
    SktUDPSend_instance(                      // Инициализация экземпляра.
        Execute:=FALSE,
        SendDat:=SendSocketDat[0]);           // Формальная запись
    SktUDPRcv_instance(                       // Инициализация экземпляра.
        Execute:=FALSE,
        RcvDat :=RcvSocketDat[0]);            // Формальная запись
    SktClose_instance(Execute:=FALSE);        // Инициализация экземпляра.
END_IF;

IF (DoSendAndRcv=TRUE) THEN
    CASE Stage OF
    1 :                                  // Запрос на создание сокета.
        SktUDPCreate_instance(
            Execute   :=TRUE,
            SrcUdpPort:=UINT#6000,             // Номер локального порта UDP
            Socket    =>WkSocket);             // Сокет

    IF (SktUDPCreate_instance.Done=TRUE) THEN
        Stage:=INT#2;                       // Нормальное завершение
    ELSIF (SktUDPCreate_instance.Error=TRUE) THEN
        Stage:=INT#10;                      // Завершение с ошибкой
    END_IF;

    2 :                                  // Запрос на передачу данных
        WkSocket.DstAdr.PortNo:=UINT#6001;
        WkSocket.DstAdr.IpAdr :='192.168.250.2';
        SktUDPSend_instance(
            Execute:=TRUE,
            Socket :=WkSocket,                 // Сокет
            SendDat:=SendSocketDat[0],         // Передаваемые данные
            Size   :=UINT#2000);               // Объем передаваемых данных

        IF (SktUDPSend_instance.Done=TRUE) THEN
                Stage:=INT#3;                       // Нормальное завершение
        ELSIF (SktUDPSend_instance.Error=TRUE) THEN
                Stage:=INT#20;                      // Завершение с ошибкой
        END_IF;

    3 :                                  // Запрос на получение данных.
```

```
        SktUDPRcv_instance(
            Execute:=TRUE,
            Socket :=WkSocket,                  // Сокет
            TimeOut:=UINT#0,                    // Время ожидания
            Size   :=UINT#2000,                 // Объем принимаемых данных
            RcvDat :=RcvSocketDat[0]);          // Принимаемые данные

        IF (SktUDPRcv_instance.Done=TRUE) THEN
            Stage:=INT#4;                      // Нормальное завершение
        ELSIF (SktUDPRcv_instance.Error=TRUE) THEN
            Stage:=INT#30;                      // Завершение с ошибкой
        END_IF;

    4 :                                 // Запрос закрытия.
        SktClose_instance(
            Execute:=TRUE,
            Socket :=WkSocket);                 // Сокет

        IF (SktClose_instance.Done=TRUE) THEN
            Stage:=INT#0;                       // Нормальное завершение
        ELSIF (SktClose_instance.Error=TRUE) THEN
            Stage:=INT#40;                      // Завершение с ошибкой
        END_IF;

    0 :                                 // Нормальное завершение
        DoSendAndRcv:=FALSE;
        Trigger     :=FALSE;

    ELSE                                // Прервано по ошибке.
        DoSendAndRcv:=FALSE;
        Trigger     :=FALSE;
    END_CASE;

END_IF;
```

Программа на удаленном узле

В рассматриваемом примере также необходимо написание программы на удаленном узле. Порядок передачи и приема данных на удаленном узле обратен порядку, который был рассмотрен выше.

1 С помощью команды SktUDPCreate запрашивается создание сокета UDP.

2 С помощью команды SktUDPRcv запрашивается прием данных. Принятые данные сохраняются в массив RcvSocketDat[].

3 С помощью команды SktUDPSend запрашивается передача данных. Передаются данные, содержащиеся в массиве SendSocketDat[].

4 Используется команда SktClose для закрытия сокета.

● Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	DoSendAndRcv	BOOL	ЛОЖЬ	Обработка
	Stage	INT	0	Смена этапа
	RcvSocketDat	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Принимаемые данные
	WkSocket	_sSOCKET	(Handle:=0, SrcAdr:=(PortNo:=0, IpAdr:=''), DstAdr:=(PortNo:=0, IpAdr:=''))	Сокет
	SendSocketDat	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Передаваемые данные
	SktUDPCreate_instance	SktUDPCreate		
	SktUDPSend_instance	SktUDPSend		
	SktUDPRcv_instance	SktUDPRcv		
	SktClose_instance	SktClose		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EIP_EtnOnlineSta	BOOL	☑	В сети

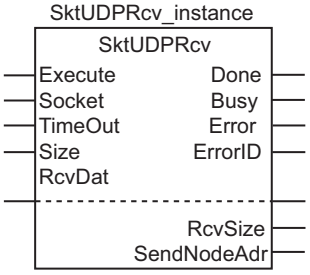
```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (DoSendAndRcv=FALSE) AND (_Eip_EtnOnlineSta=TRUE) ) THEN
    DoSendAndRcv:=TRUE;
    Stage       :=INT#1;
    SktUDPCreate_instance(Execute:=FALSE);  // Инициализация экземпляра.
    SktUDPSend_instance(                    // Инициализация экземпляра.
        Execute:=FALSE,
        SendDat:=SendSocketDat[0]);         // Формальная запись
    SktUDPRcv_instance(                     // Инициализация экземпляра.
        Execute:=FALSE,
        RcvDat :=RcvSocketDat[0]);          // Формальная запись
    SktClose_instance(Execute:=FALSE);      // Инициализация экземпляра.
END_IF;

IF (DoSendAndRcv=TRUE) THEN
    CASE Stage OF
    1 :                                 // Запрос на создание сокета.
        SktUDPCreate_instance(
            Execute   :=TRUE,
            SrcUdpPort:=UINT#6001,            // Номер локального порта UDP
```

```
        Socket    =>WkSocket);            // Сокет

    IF (SktUDPCreate_instance.Done=TRUE) THEN
        Stage:=INT#2;                     // Нормальное завершение
    ELSIF (SktUDPCreate_instance.Error=TRUE) THEN
        Stage:=INT#10;                    // Завершение с ошибкой
    END_IF;
2 :                                 // Запрос на получение данных
    SktUDPRcv_instance(
        Execute:=TRUE,
        Socket :=WkSocket,                // Сокет
        TimeOut:=UINT#0,                  // Время ожидания
        Size   :=UINT#2000,               // Объем принимаемых данных
        RcvDat :=RcvSocketDat[0]);        // Принимаемые данные

    IF (SktUDPRcv_instance.Done=TRUE) THEN
        Stage:=INT#3;                     // Нормальное завершение
    ELSIF (SktUDPRcv_instance.Error=TRUE) THEN
        Stage:=INT#20;                    // Завершение с ошибкой
    END_IF;

3 :                                 // Запрос на передачу данных.
    WkSocket.DstAdr.PortNo:=UINT#6000;
    WkSocket.DstAdr.IpAdr :='192.168.250.1';
    SktUDPSend_instance(
        Execute:=TRUE,
    Socket :=WkSocket,                // Сокет
    SendDat:=SendSocketDat[0],        // Передаваемые данные
    Size   :=UINT#2000);              // Объем передаваемых данных

IF (SktUDPSend_instance.Done=TRUE) THEN
    Stage:=INT#4;                     // Нормальное завершение
ELSIF (SktUDPSend_instance.Error=TRUE) THEN
    Stage:=INT#30;                    // Завершение с ошибкой
END_IF;

4 :                                 // Запрос закрытия.
    SktClose_instance(
        Execute:=TRUE,
        Socket :=WkSocket);               // Сокет

    IF (SktClose_instance.Done=TRUE) THEN
        Stage:=INT#0;                     // Нормальное завершение
    ELSIF (SktClose_instance.Error=TRUE) THEN
        Stage:=INT#40;                    // Завершение с ошибкой
    END_IF;

0 :                                 // Нормальное завершение
    DoSendAndRcv:=FALSE;
    Trigger     :=FALSE;
```

```
    ELSE                                    // Прервано по ошибке.
        DoSendAndRcv:=FALSE;
        Trigger     :=FALSE;
    END_CASE;

END_IF;
```

SktUDPRcv

Команда SktUDPRcv считывает данные из буфера приема для сокета UDP на встроенном порте EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktUDPRcv	Прием через сокет UDP	FB		SktUDPRcv_instance(Execute, Socket, TimeOut, Size, RcvDat, Done, Busy, Error, ErrorID, RcvSize, SendNodeAdr);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---
TimeOut	Время ожидания		0: время ожидания не задано 1...65535: 0,1...6553,5 с	Зависит от типа данных.	0,1 с	0
Size	Сохраняемый объем		Количество байтов данных, которые нужно прочитать из буфера приема	0…2000	Байты	1
RcvDat[] (массив)	Принимаемые данные	Вход-выход	Принимаемые данные	Зависит от типа данных.	---	---
RcvSize	Объем принятых данных	Выход	Количество байтов данных, фактически сохраненных в RcvDat[]	0…2000	Байты	---
SendNodeAdr	Адрес узла-источника		Адрес узла-источника	---	---	

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1237.																			
TimeOut							ОК													
Size							ОК													

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RcvDat[] (массив)		ОК																		
RcvSize							ОК													
SendNodeAdr	Подробные сведения о структуре _sSOCKET_ADDRESS см. в разделе *Функция* на стр. 2-1237.																			

Функция

Команда SktUDPRcv считывает данные из буфера приема, относящегося к сокету, который указан параметром *Socket*, и сохраняет их в переменную RcvDat[] (принятые данные). Количество байтов данных, которые нужно сохранить, указывается в параметре *Size*.
Фактическое количество сохраненных байтов данных выводится в выходную переменную *RcvSize*.
Адрес узла, передавшего данные, сохраняется в переменную *SendNodeAdr*.
Если данных в буфере приема нет, команда ожидает поступления данных в течение времени ожидания, заданного параметром *TimeOut*.
Если выполнение команды завершается нормально, значение переменной *Done* меняется на ИСТИНА.
Принятые данные сохраняются в переменную RcvDat[], если команда завершается нормально.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор для обмена данными	UDINT	Зависит от типа данных.	---	0
SrcAdr*1	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''
DstAdr*1	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''

*1. Эти члены структуры для данной команды не используются.

Для переменной *SendNodeAdr* используется структурный тип данных _sSOCKET_ADDRESS. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
SendNodeAdr	Адрес узла-источника	Адрес узла-источника	_sSOCKET_ADDRESS	---	---	---
PortNo	Номер порта	Номер порта UPD узла, передавшего данные	UINT	1…65535	---	---
IpAdr	IP-адрес	IP-адрес узла, передавшего данные	STRING	Зависит от типа данных.		

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.

*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX. Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.

*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

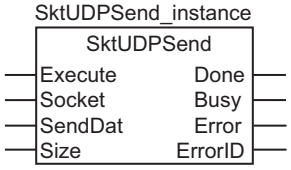
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- С помощью одной команды можно прочитать до 2000 байтов данных из буфера приема.
- Если объем данных, полученных указанным сокетом, меньше значения в *Size*, в RecDat[] сохраняются все полученные данные. В этом случае в *RcvSize* записывается фактический объем сохраненных данных.
- Если же объем данных, полученных указанным сокетом, больше значения в *Size*, в RecDat[] сохраняются полученные данные в том объеме, который указан параметром *Size*.
- Если *Size* = 0, чтение полученных данных не производится.
- Если команда SktClose закрывает соединение, когда в буфере приема нет данных, происходит нормальное завершение без ожидания получения данных, даже если время ожидания еще не истекло. В этом случае *RcvSize* = 0.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Имеется ошибка в настройке локального IP-адреса.
  b) Через сокет, указанный параметром *Socket*, в данный момент принимаются данные.
  c) Сокет, указанный параметром *Socket*, не открыт.
  d) Дескриптора, указанного параметром *Socket.Handle*, не существует.

Пример программы

См. *Пример программы* на стр. 2-1229 для команды SktUDPCreate.

SktUDPSend

Команда SktUDPSend передает данные с порта UDP на встроенном порте EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktUDPSend	Передача через сокет UDP	FB		SktUDPSend_instance(Execute, Socket, SendDat, Size, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---
SendDat[] (массив)	Передаваемые данные		Передаваемые данные	Зависит от типа данных.		
Size	Объем передаваемых данных		Объем передаваемых данных	0…2000	Байты	1

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1240.																			
SendDat[] (массив)		ОК																		
Size							ОК													

Функция

Команда SktUDPSend передает содержимое SendDat[] (передаваемые данные) через сокет, который указан параметром *Socket*.
Количество байтов данных, которые нужно передать, указывается в параметре *Size*.
Удаленный узел указывается с помощью параметра *Socket.DstAdr*.
Если выполнение команды завершается нормально, значение переменной *Done* меняется на ИСТИНА.
Содержимое SendDat[] передается в буфер передачи, если команда завершается нормально.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор для обмена данными	UDINT	Зависит от типа данных.	---	0
SrcAdr*1	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''
DstAdr	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''

*1. Эти члены структуры для данной команды не используются.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

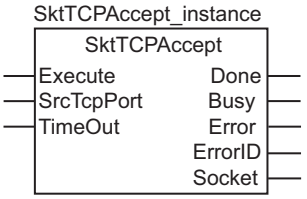
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- С помощью одной команды можно передать до 2000 байтов данных. Даже если массив SendDat[] содержит больше 2000 байтов данных, передается максимум 2000 байтов. В случае указания широковещательного адреса может быть передано только 1472 байта.
- Если *Size* = 0, по интерфейсу связи передается 0 байт данных.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Имеется ошибка в настройке локального IP-адреса.
  b) Значение члена структуры в *Socket* находится за пределами допустимого диапазона.
  c) Через сокет, указанный параметром *Socket*, в данный момент передаются данные.
  d) Сокет, указанный параметром *Socket*, не открыт.
  e) Не удалось разрешить адрес удаленного узла, указанный в параметре *Socket* с использованием имени домена.
  f) Дескриптора, указанного параметром *Socket.Handle*, не существует.
  g) Значение *Size* превышает количество элементов массива SendDat[].

Пример программы

См. *Пример программы* на стр. 2-1229 для команды SktUDPCreate.

SktTCPAccept

Команда SktTCPAccept запрашивает принятие сокета TCP для встроенного порта EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktTCPAccept	Принятие сокета TCP	FB		SktTCPAccept_instance(Execute, SrcTcpPort, TimeOut, Done, Busy, Error, ErrorID, Socket);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
SrcTcpPort	Номер локального порта TCP.	Вход	Номер локального порта TCP.	1…65535	---	1
TimeOut	Время ожидания		0: время ожидания не задано 1...65535: 0,1...6553,5 с	Зависит от типа данных.	0,1 с	0
Socket	Сокет	Выход	Сокет	---	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SrcTcpPort							OK													
TimeOut							OK													
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1243.																			

Функция

Команда SktTCPAccept запрашивает принятие порта, указанного параметром *ScrTcpPort* (номер локального порта TCP). Для этого она выполняет функции сокетов Socket(), Bind(), Listen() и Accept().

Команда ожидает установления соединения с удаленным узлом в течение времени ожидания, заданного параметром *TimeOut*.

Если выполнение команды завершается нормально, значение переменной *Done* меняется на ИСТИНА.

Соединение устанавливается, если команда завершается нормально.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор для обмена данными	UDINT	Зависит от типа данных.	---	0
SrcAdr	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''
DstAdr	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''

*1. В этот член структуры выводится пустой символ (NULL).

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

- Дополнительные сведения о службах сокетов см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.
- Эту команду можно выполнить несколько раз для открытия соединений с несколькими клиентами с использованием одного номера локального порта. Для каждого соединения возвращается другой сокет.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Для закрытия дескрипторов, созданных с помощью этой команды, используйте команду SktClose.
- При переходе в режим «Программирование» созданные этой командой дескрипторы деактивируются.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Имеется ошибка в настройке локального IP-адреса.
  b) Значение *SrcTcpPort* находится за пределами допустимого диапазона.
  c) Для сокета, указанного параметром *SrcTcpPort*, в данный момент выполняется операция открытия.
  d) Для сокета, указанного параметром *SrcTcpPort*, в данный момент выполняется операция закрытия.
  e) Соединение не было открыто в течение времени ожидания, указанного в *TimeOut*.

Информация о версии

- Количество сокетов, которые могут быть открыты одновременно, зависит от версии модуля ЦПУ, как показано в следующей таблице. Это общие предельные значения в сумме для сокетов UDP и TCP.

Версия модуля ЦПУ	Количество сокетов
1.03 или более поздняя версия	Макс. 30*1
1.02 или более ранняя версия	Макс. 16

*1. В случае модулей ЦПУ NX102 может быть открыто максимум 60 сокетов.

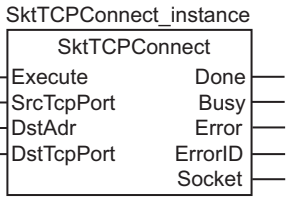
- При использовании модуля ЦПУ версии 1.10 или более поздней версии значение *Socket* не изменяется, даже если значение *Error* меняется на ИСТИНА. В случае версии 1.09 или более ранней версии значение *Socket* меняется на *0*.

Пример программы

См. *Пример программы* на стр. 2-1249 для команды SktTCPConnect.

SktTCPConnect

Команда SktTCPConnect устанавливает соединение с удаленным портом TCP через встроенный порт EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktTCPConnect	Соединение с сокетом TCP	FB		SktTCPConnect_instance(Execute , SrcTcpPort, DstAdr, DstTcpPort, Done, Busy, Error, ErrorID, Socket);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
SrcTcpPort	Номер локального порта TCP.	Вход	Номер локального порта TCP. Если указано значение 0, автоматически назначается доступный порт TCP с номером 1024 или больше. Хорошо известные номера портов при этом не назначаются.	Зависит от типа данных.	---	0
DstAdr	Адрес назначения		IP-адрес назначения или имя хоста	Макс. 200 байт		---
DstTcpPort	Номер удаленного порта TCP		Номер удаленного порта TCP	1…65 535		1
Socket	Сокет	Выход	Сокет	---	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SrcTcpPort							ОК													
DstAdr																				ОК
DstTcpPort							ОК													
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1247.																			

Функция

Команда SktTCPConnect запрашивает установление соединения между локальным портом TCP с номером *SrcTcpPort* и удаленным (адресуемым) портом TCP с номером *DstTcpPort* по адресу назначения *DstAdr*. Для этого она выполняет функцию сокетов Connect().
Если выполнение команды завершается нормально, значение переменной *Done* меняется на ИСТИНА.
Соединение устанавливается, если команда завершается нормально.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор для обмена данными	UDINT	Зависит от типа данных.	---	0
SrcAdr	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''
DstAdr	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''

*1. В этот член структуры выводится пустой символ (NULL).

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Для закрытия дескрипторов, созданных с помощью этой команды, используйте команду SktClose.
- При переходе в режим «Программирование» созданные этой командой дескрипторы деактивируются.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Имеется ошибка в настройке локального IP-адреса.
  b) Значение *DstAdr* находится за пределами допустимого диапазона.
  c) Значение *DstTcpPort* находится за пределами допустимого диапазона.
  d) Порт TCP, указанный в переменной *SrcTcpPort*, уже открыт.
  e) Удаленного узла, указанного в переменной *DstAdr*, не существует.
  f) Удаленный узел, указанный в переменных *DstAdr* и *DstTcpPort*, не ожидает установления соединения.
  g) Не удалось разрешить адрес для имени хоста, указанного в переменной *DstAdr*.
  h) Соединение уже открыто для того же клиента (IP-адрес и порт TCP).

Информация о версии

- Количество сокетов, которые могут быть открыты одновременно, зависит от версии модуля ЦПУ, как показано в следующей таблице. Это общие предельные значения в сумме для сокетов UDP и TCP.

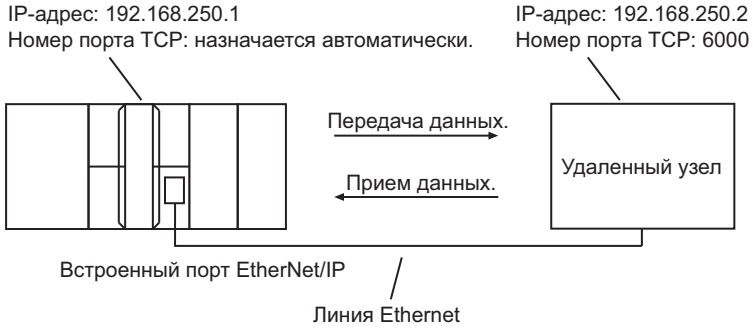
Версия модуля ЦПУ	Количество сокетов
1.03 или более поздняя версия	Макс. 30*1
1.02 или более ранняя версия	Макс. 16

*1. В случае модулей ЦПУ NX102 может быть открыто максимум 60 сокетов.

- При использовании модуля ЦПУ версии 1.10 или более поздней версии значение *Socket* не изменяется, даже если значение *Error* меняется на ИСТИНА. В случае версии 1.09 или более ранней версии значение *Socket* меняется на *0*.

Пример программы

Рассмотрим пример программы для случая, когда для обмена данными между Серия NJ/NX, модули ЦПУ и удаленным узлом используется служба сокета TCP.



Программа пользователя Серия NJ/NX, модули ЦПУ

Соблюдается следующий порядок действий:

1 С помощью команды SktTCPConnect запрашивается установление соединения с портом TCP на удаленном узле.

2 Используется команда SktClearBuf для очистки буфера приема для сокета TCP.

3 Используется команда SktGetTCPStatus для чтения состояния сокета TCP.

4 С помощью команды SktTCPSend запрашивается передача данных. Передаются данные, содержащиеся в массиве SendSocketDat[].

5 С помощью команды SktTCPRcv запрашивается прием данных. Принятые данные сохраняются в массив RcvSocketDat[].

6 Используется команда SktClose для закрытия сокета.

● Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	DoTCP	BOOL	ЛОЖЬ	Обработка
	Stage	INT	0	Смена этапа
	RcvSocketDat	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Принимаемые данные
	WkSocket	_sSOCKET	(Handle:=0, SrcAdr:=(PortNo:=0, IpAdr:=''), DstAdr:=(PortNo:=0, IpAdr:=''))	Сокет

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	SendSocketDat	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Передаваемые данные
	SktTCPConnect_instance	SktTCPConnect		
	SktClearBuf_instance	SktClearBuf		
	SktGetTCPStatus_instance	SktGetTCPStatus		
	SktTCPSend_instance	SktTCPSend		
	SktTCPRcv_instance	SktTCPRcv		
	SktClose_instance	SktClose		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EIP_EtnOnlineSta	BOOL	☑	В сети

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (DoTCP=FALSE) AND (_Eip_EtnOnlineSta=TRUE) ) THEN
    DoTCP:=TRUE;
    Stage:=INT#1;
    SktTCPConnect_instance(Execute:=FALSE);  // Инициализация экземпляра.
    SktClearBuf_instance(Execute:=FALSE);  // Инициализация экземпляра.
    SktGetTCPStatus_instance(Execute:=FALSE);  // Инициализация экземпляра.
    SktTCPSend_instance(  // Инициализация экземпляра.
        Execute:=FALSE,
        SendDat:=SendSocketDat[0]);  // Формальная запись
    SktTCPRcv_instance(  // Инициализация экземпляра..
        Execute:=FALSE,
        RcvDat :=RcvSocketDat[0]);  // Формальная запись
    SktClose_instance(Execute:=FALSE);  // Инициализация экземпляра.
END_IF;

IF (DoTCP=TRUE) THEN
    CASE Stage OF
    1 :  // Запрос установления соединения.
        SktTCPConnect_instance(
            Execute   :=TRUE,
            SrcTcpPort:=UINT#0,  // Номер локального порта TCP: назначается автомат
ически.
            DstAdr    :='192.168.250.2',  // Удаленный IP-адрес
            DstTcpPort:=UINT#6000,  // Номер удаленного порта TCP
            Socket    =>WkSocket);  // Сокет

        IF (SktTCPConnect_instance.Done=TRUE) THEN
            Stage:=INT#2;  // Нормальное завершение
        ELSIF (SktTCPConnect_instance.Error=TRUE) THEN
            Stage:=INT#10;  // Завершение с ошибкой
        END_IF;
```

```
2 :  // Очистить буфер приема.
    SktClearBuf_instance(
        Execute:=TRUE,
        Socket :=WkSocket);  // Сокет

    IF (SktClearBuf_instance.Done=TRUE) THEN
        Stage:=INT#3;  // Нормальное завершение
    ELSIF (SktClearBuf_instance.Error=TRUE) THEN
        Stage:=INT#20;  // Завершение с ошибкой
    END_IF;

3 :  // Запрос чтения состояния.
    SktGetTCPStatus_instance(
        Execute:=TRUE,
        Socket :=WkSocket);  // Сокет

    IF (SktGetTCPStatus_instance.Done=TRUE) THEN
        Stage:=INT#4;  // Нормальное завершение
    ELSIF (SktGetTCPStatus_instance.Error=TRUE) THEN
        Stage:=INT#30;  // Завершение с ошибкой
    END_IF;

4 :  // Запрос на передачу данных
    SktTCPSend_instance(
        Execute:=TRUE,
        Socket :=WkSocket,  // Сокет
        SendDat:=SendSocketDat[0],  // Передаваемые данные
        Size   :=UINT#2000);  // Объем передаваемых данных

    IF (SktTCPSend_instance.Done=TRUE) THEN
        Stage:=INT#5;  // Нормальное завершение
    ELSIF (SktTCPSend_instance.Error=TRUE) THEN
        Stage:=INT#40;  // Завершение с ошибкой
    END_IF;

5 :  // Запрос на получение данных
    SktTCPRcv_instance(
        Execute:=TRUE,
        Socket :=WkSocket,  // Сокет
        TimeOut:=UINT#0,  // Время ожидания
        Size   :=UINT#2000,  // Объем принимаемых данных
        RcvDat :=RcvSocketDat[0]);  // Принимаемые данные

    IF (SktTCPRcv_instance.Done=TRUE) THEN
        Stage:=INT#6;  // Нормальное завершение
    ELSIF (SktTCPRcv_instance.Error=TRUE) THEN
        Stage:=INT#50;  // Завершение с ошибкой
    END_IF;

6 :  // Запрос закрытия.
    SktClose_instance(
```

```
            Execute:=TRUE,
            Socket :=WkSocket);  // Сокет

        IF (SktClose_instance.Done=TRUE) THEN
            Stage:=INT#0;  // Нормальное завершение
        ELSIF (SktClose_instance.Error=TRUE) THEN
            Stage:=INT#40;  // Завершение с ошибкой
        END_IF;
    0 :  // Нормальное завершение
        DoTCP  :=FALSE;
        Trigger:=FALSE;

    ELSE  // Прервано по ошибке.
        DoTCP  :=FALSE;
        Trigger:=FALSE;
    END_CASE;

END_IF;
```

Программа на удаленном узле

В рассматриваемом примере также необходимо написание программы на удаленном узле. Порядок передачи и приема данных на удаленном узле обратен порядку, который был рассмотрен выше.

1 С помощью команды SktTCPAccept запрашивается принятие сокета TCP.

2 С помощью команды SktTCPRcv запрашивается прием данных. Принятые данные сохраняются в массив RcvSocketDat[].

3 С помощью команды SktTCPSend запрашивается передача данных. Передаются данные, содержащиеся в массиве SendSocketDat[].

4 Используется команда SktClose для закрытия сокета.

● Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	DoTCP	BOOL	ЛОЖЬ	Обработка
	Stage	INT	0	Смена этапа
	RcvSocketDat	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Принимаемые данные
	WkSocket	_sSOCKET	(Handle:=0, SrcAdr:=(PortNo:=0, IpAdr:=''), DstAdr:=(PortNo:=0, IpAdr:=''))	Сокет

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	SendSocketDat	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Передаваемые данные
	SktTCPAccept_instance	SktTCPAccept		
	SktTCPSend_instance	SktTCPSend		
	SktTCPRcv_instance	SktTCPRcv		
	SktClose_instance	SktClose		

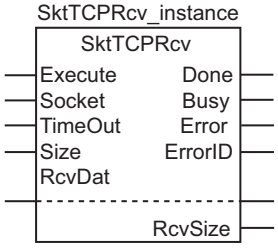
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EIP_EtnOnlineSta	BOOL	☑	В сети

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (DoTCP=FALSE) AND (_Eip_EtnOnlineSta=TRUE) ) THEN
    DoTCP:=TRUE;
    Stage:=INT#1;
    SktTCPAccept_instance(Execute:=FALSE);    // Инициализация экземпляра.
    SktTCPSend_instance(                      // Инициализация экземпляра.
        Execute:=FALSE,
        SendDat:=SendSocketDat[0]);           // Формальная запись
    SktTCPRcv_instance(                       // Инициализация экземпляра.
        Execute:=FALSE,
        RcvDat :=RcvSocketDat[0]);            // Формальная запись
    SktClose_instance(Execute:=FALSE);        // Инициализация экземпляра.
END_IF;

IF (DoTCP=TRUE) THEN
    CASE Stage OF
    1 :                                 // Запрос на принятие соединения с сокетом.
        SktTCPAccept_instance(
            Execute   :=TRUE,
            SrcTcpPort:=UINT#6000,          // Номер локального порта TCP
            TimeOut   :=UINT#0,             // Время ожидания
            Socket    =>WkSocket);          // Сокет

        IF (SktTCPAccept_instance.Done=TRUE) THEN
            Stage:=INT#2;                    // Нормальное завершение
        ELSIF (SktTCPAccept_instance.Error=TRUE) THEN
            Stage:=INT#10;                   // Завершение с ошибкой
        END_IF;
    2 :                                 // Запрос на получение данных
        SktTCPRcv_instance(
            Execute:=TRUE,
            Socket :=WkSocket,               // Сокет
            TimeOut:=UINT#0,                 // Время ожидания
            Size   :=UINT#2000,              // Объем принимаемых данных
```

```
            RcvDat :=RcvSocketDat[0]);          // Принимаемые данные

        IF (SktTCPRcv_instance.Done=TRUE) THEN
            Stage:=INT#3;                       // Нормальное завершение
        ELSIF (SktTCPRcv_instance.Error=TRUE) THEN
            Stage:=INT#20;                      // Завершение с ошибкой
        END_IF;

    3 :                                 // Запрос на передачу данных.
        SendSocketDat:=RcvSocketDat;
        SktTCPSend_instance(
            Execute:=TRUE,
            Socket :=WkSocket,                  // Сокет
            SendDat:=SendSocketDat[0],          // Передаваемые данные
            Size   :=UINT#2000);                // Объем передаваемых данных

        IF (SktTCPSend_instance.Done=TRUE) THEN
            Stage:=INT#4;                       // Нормальное завершение
        ELSIF (SktTCPSend_instance.Error=TRUE) THEN
            Stage:=INT#30;                      // Завершение с ошибкой
        END_IF;

    4 :                                 // Запрос закрытия.
        SktClose_instance(
            Execute:=TRUE,
            Socket :=WkSocket);                 // Сокет

        IF (SktClose_instance.Done=TRUE) THEN
            Stage:=INT#0;                       // Нормальное завершение
        ELSIF (SktClose_instance.Error=TRUE) THEN
            Stage:=INT#40;                      // Завершение с ошибкой
        END_IF;

    0 :                                 // Нормальное завершение
        DoTCP  :=FALSE;
        Trigger:=FALSE;

    ELSE                                // Прервано по ошибке.
        DoTCP  :=FALSE;
        Trigger:=FALSE;
    END_CASE;

END_IF;
```

SktTCPRcv

Команда SktTCPRcv считывает данные из буфера приема для сокета TCP на встроенном порте EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktTCPRcv	Прием через сокет TCP	FB		SktTCPRcv_instance(Execute, Socket, TimeOut, Size, RcvDat, Done, Busy, Error, ErrorID,RcvSize);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---
TimeOut	Время ожидания		0: время ожидания не задано 1...65535: 0,1...6553,5 с	Зависит от типа данных.	0,1 с	0
Size	Сохраняемый объем		Количество байтов данных, которые нужно прочитать из буфера приема	0…2000	Байты	1
RcvDat[] (массив)	Принимаемые данные	Вход-выход	Принимаемые данные	Зависит от типа данных.	---	---
RcvSize	Объем принятых данных	Выход	Количество байтов данных, фактически сохраненных в RcvDat[]	0…2000	Байты	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1256.																			
TimeOut							ОК													
Size							ОК													
RcvDat[] (массив)		ОК																		
RcvSize							ОК													

Функция

Команда SktTCPRcv считывает данные из буфера приема, относящегося к сокету, который указан параметром *Socket*, и сохраняет их в переменную RcvDat[] (принятые данные). Количество байтов данных, которые нужно сохранить, указывается в параметре *Size*.
Фактическое количество сохраненных байтов данных выводится в выходную переменную *RcvSize*.
Если данных в буфере приема нет, команда ожидает поступления данных в течение времени ожидания, заданного параметром *TimeOut*.
Если выполнение команды завершается нормально, значение переменной *Done* меняется на ИСТИНА.
Принятые данные сохраняются в переменную RcvDat[], если команда завершается нормально.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор для обмена данными	UDINT	Зависит от типа данных.	---	0
SrcAdr*1	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''
DstAdr*1	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''

*1. Эти члены структуры для данной команды не используются.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- С помощью одной команды можно прочитать до 2000 байтов данных. Даже если массив RcvDat[] содержит больше 2000 байтов данных, считывается максимум 2000 байтов.
- Если объем данных, полученных указанным сокетом, меньше значения в *Size*, в RecDat[] сохраняются все полученные данные. В этом случае в *RcvSize* записывается фактический объем сохраненных данных.
- Если же объем данных, полученных указанным сокетом, больше значения в *Size*, в RecDat[] сохраняются полученные данные в том объеме, который указан параметром *Size*.
- Если *Size* = 0, чтение полученных данных не производится.
- Если команда SktClose закрывает соединение, когда в буфере приема нет данных, происходит завершение с ошибкой, даже если время ожидания еще не истекло.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Имеется ошибка в настройке локального IP-адреса.
  b) Значение члена структуры в *Socket* находится за пределами допустимого диапазона.
  c) Через сокет, указанный параметром *Socket*, в данный момент принимаются данные.
  d) Сокет, указанный параметром *Socket*, не подключен.
  e) Дескриптора, указанного параметром *Socket.Handle*, не существует.

f) Данные не были получены за время ожидания, заданное параметром *TimeOut*.
g) Сокет был закрыт командой SktClose.

Пример программы

См. *Пример программы* на стр. 2-1249 для команды SktTCPConnect.

SktTCPSend

Команда SktTCPSend передает данные с порта TCP на встроенном порте EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktTCPSend	Передача через сокет TCP	FB	SktTCPSend_instance SktTCPSend Execute — Done Socket — Busy SendDat — Error Size — ErrorID	SktTCPSend_instance(Execute, Socket, SendDat, Size, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---
SendDat[] (массив)	Передаваемые данные		Передаваемые данные	Зависит от типа данных.		
Size	Объем передаваемых данных		Объем передаваемых данных	0…2000	Байты	1

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1259.																			
SendDat[] (массив)		ОК																		
Size							ОК													

Функция

Команда SktTCPSend передает содержимое SendDat[] (передаваемые данные) через сокет, который указан параметром *Socket*.
Количество байтов данных, которые нужно передать, указывается в параметре *Size*.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор для обмена данными	UDINT	Зависит от типа данных.	---	0
SrcAdr*1	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''
DstAdr*1	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''

*1. Эти члены структуры для данной команды не используются.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

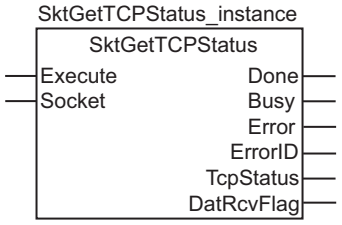
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- С помощью одной команды можно передать до 2000 байтов данных. Даже если массив SendDat[] содержит больше 2000 байтов данных, передается максимум 2000 байтов.
- Если *Size* = 0, данные не передаются.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Имеется ошибка в настройке локального IP-адреса.
  b) Значение члена структуры в *Socket* находится за пределами допустимого диапазона.
  c) Через сокет, указанный параметром *Socket*, в данный момент передаются данные.
  d) Сокет, указанный параметром *Socket*, не подключен.
  e) Дескриптора, указанного параметром *Socket.Handle*, не существует.

Пример программы

См. *Пример программы* на стр. 2-1249 для команды SktTCPConnect.

SktGetTCPStatus

Команда SktGetTCPStatus производит чтение состояния сокета TCP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktGetTCPStatus	Чтение состояния сокета TCP	FB		SktGetTCPStatus_instance(Execute, Socket, Done, Busy, Error, ErrorID, TcpStatus, DatRcvFlag);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---
TcpStatus	Состояние соединения TCP	Выход	Состояние соединения TCP	*1	---	---
DatRcvFlag	Флаг «Данные приняты»		ИСТИНА: Данные приняты. ЛОЖЬ: Данные не приняты.	Зависит от типа данных.		

*1. _CLOSED, _LISTEN, _SYN_SENT, _SYN_RECEIVED, _ESTABLISHED, _CLOSE_WAIT, _FIN_WAIT1, _CLOSING, _LAST_ACK, _FIN_WAIT2, or _TIME_WAIT

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1262.																			
TcpStatus	Сведения о перечислителях перечислимого типа _eCONNECTION_STATE см. в разделе *Функция* на стр. 2-1262.																			
DatRcvFlag	ОК																			

Функция

Команда SktGetTCPStatus возвращает состояние *TcpStatus* соединения TCP для сокета, который указан параметром *Socket*.
Если в буфере приема имеются принятые данные, состояние флага «Данные приняты» *DatRecvFlag* меняется на ИСТИНА.
Если выполнение команды завершается нормально, значение переменной *Done* меняется на ИСТИНА.

Данные сохраняются в переменные *TcpStatus* и *DatRcvFlag*, если команда завершается нормально.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор для обмена данными	UDINT	Зависит от типа данных.	---	0
SrcAdr*1	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''
DstAdr*1	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''

*1. Эти члены структуры для данной команды не используются.

Для переменной *TcpStatus* используется перечислимый тип данных _eCONNECTION_STATE. Каждый перечислитель указывает на определенное состояние TCP. Значения перечислителей и соответствующие им состояния TCP приведены в таблице ниже.

Перечислители	Состояние TCP	Описание
_CLOSED	CLOSED	Соединение закрыто.
_LISTEN	LISTEN	Сервер ожидает запроса на соединение (SYN) с открытием в пассивном режиме.
_SYN_SENT	SYN SENT	Клиент отправил запрос на соединение (SYN) для открытия в активном режиме и ожидает подтверждения (SYN + ACK).
_SYN_RECEIVED	SYN RECEIVED	Сервер отправил подтверждение (SYN + ACK) в ответ на запрос соединения (SYN) и ожидает подтверждения (ACK).
_ESTABLISHED	ESTABLISHED	Соединение установлено.
_CLOSE_WAIT	CLOSE WAIT	Сервер отправил подтверждение (ACK) на запрос на закрытие соединения (FIN) и ждет, когда серверное приложение будет готово к закрытию.
_FIN_WAIT1	FIN WAIT-1	Клиент отправил запрос на закрытие соединения (FIN) и ожидает подтверждения (ACK).
_CLOSING	CLOSING	Клиент и сервер одновременно получили запрос на закрытие соединения (FIN) и ждут подтверждения (ACK).

Перечислители	Состояние TCP	Описание
_LAST_ACK	LAST-ACK	Сервер отправил запрос на закрытие соединения (FIN) и ожидает подтверждения (ACK).
_FIN_WAIT2	FIN WAIT-2	Клиент ожидает запроса на закрытие соединения (FIN).
_TIME_WAIT	TIME WAIT	Клиент получил подтверждение (ACK) на запрос на закрытие соединения (FIN) и ожидает завершения процесса на сервере.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

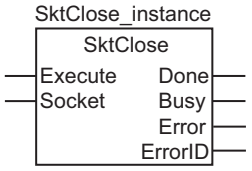
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение члена структуры в *Socket* находится за пределами допустимого диапазона.
  b) Дескриптора, указанного параметром *Socket.Handle*, не существует.

Пример программы

См. *Пример программы* на стр. 2-1249 для команды SktTCPConnect.

SktClose

Команда SktClose закрывает указанный сокет TCP или UDP для встроенного порта EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktClose	Закрытие сокета TCP/UDP	FB		SktClose_instance(Execute, Socket, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1265.																			

Функция

Команда SktClose закрывает сокет, указанный параметром *Socket*.
Если указан сокет TCP, перед его закрытием производится отсоединение.
Если дескриптор сокета *Socket.Handle* = 0, будут закрыты все порты TCP и UDP, которые в настоящее время используют службу сокета.
Если выполнение команды завершается нормально, значение переменной *Done* меняется на ИСТИНА.
Операция закрытия для сокетов TCP и UDP выполняется, если команда завершается нормально.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор закрываемого соединения. 0: закрыть все соединения TCP, которые в настоящее время используют службу сокета.	UDINT	Зависит от типа данных.	---	0
SrcAdr*1	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''
DstAdr*1	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''

*1. Эти члены структуры для данной команды не используются.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

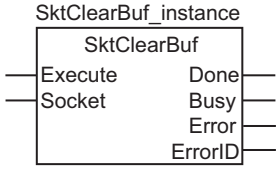
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Если выполняется команда SktUDPRcv или SktTCPRcv, а затем выполняется команда SktClose, когда сокет для указанного дескриптора находится в режиме ожидания для приема данных, состояние ожидания отменяется.
- Если для одного и того же номера локального порта открыто несколько соединений, закрывается только соединение для указанного сокета.
- Если значение дескриптора сокета *Socket.Handle* = 0, все соединения, находящиеся в режиме ожидания для команды SktTCPAccept, будут отменены.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Имеется ошибка в настройке локального IP-адреса.
  b) Значение члена структуры в *Socket* находится за пределами допустимого диапазона.
  c) Дескриптора, указанного параметром *Socket.Handle*, не существует.

Пример программы

См. *Пример программы* на стр. 2-1229 для команды SktUDPCreate и *Пример программы* на стр. 2-1249 для команды SktTCPConnect.

SktClearBuf

Команда SktClearBuf очищает буфер приема для указанного сокета TCP или UDP для встроенного порта EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktClearBuf	Очистка буфера приема сокета TCP/UDP	FB		SktClearBuf_instance(Execute, Socket, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1268.																			

Функция

Команда SktClearBuf очищает буфер приема для сокета, который указан параметром *Socket*. Если выполнение команды завершается нормально, значение переменной *Done* меняется на ИСТИНА.
Операция очистки для буфера приема выполняется, если команда завершается нормально.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор сокета, для которого нужно очистить буфер приема.	UDINT	Зависит от типа данных.	---	0
SrcAdr*1	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''
DstAdr*1	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''

*1. Эти члены структуры для данной команды не используются.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение члена структуры в *Socket* находится за пределами допустимого диапазона.
  b) Сокета, указанного в переменной *Socket*, не существует.
  c) Дескриптора, указанного параметром *Socket.Handle*, не существует.

Пример программы

См. *Пример программы* на стр. 2-1249 для команды SktTCPConnect.

SktSetOption

Команда SktSetOption позволяет настроить дополнительный параметр для указанного сокета TCP для встроенного порта EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SktSetOption	Настройка дополнительного параметра сокета TCP	FB	SktSetOption_instance SktSetOption Execute Done Socket Busy OptionType Error OptionParam ErrorID	SktSetOption_instance(Execute, Socket, OptionType, OptionParam, Done, Busy, Error, ErrorID);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.12 или более поздней и Sysmac Studio версии 1.16 или выше.
В случае модуля ЦПУ NX1P2 для использования этой команды требуются версия модуля 1.14 или более поздняя и Sysmac Studio версии 1.18 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---
OptionType	Тип дополнительного параметра		Тип дополнительного параметра сокета	---	---	---
OptionParam	Настройка дополнительного параметра		Настройка в соответствии с указанным дополнительным параметром сокета	---	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Функция* на стр. 2-1272.																			
OptionType	Сведения о перечислителях перечислимого типа _eSKT_OPTION_TYPE см. в разделе *Функция* на стр. 2-1272.																			
OptionParam	OK *1																			

*1. Вводить константу (литерал) нельзя. Следует указать переменную.

Функция

Команда SktSetOption позволяет настроить дополнительный параметр для сокета, указанного в переменной *Socket*.
Если выполнение команды завершается нормально, значение *Done* меняется на ИСТИНА.
Дополнительный параметр сокета будет настроен, если выполнение команды завершится нормально.

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор сокета, для которого нужно очистить буфер приема.	UDINT	Зависит от типа данных.	---	0
SrcAdr*1	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65 535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''
DstAdr*1	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65 535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		''

*1. Эти члены структуры для данной команды не используются.

В следующей таблице приведены: значение параметра *OptionType*, которое можно указать, а также тип данных параметра *OptionParam*, который можно выбрать для указанного параметра *OptionType*. Также приведено значение по умолчанию, которое используется для работы, когда эта команда не используется.

OptionType		OptionParam		
Перечислитель	Значение	Возможный тип данных	Смысл значения	По умолчанию
_TCP_NODELAY	Указывает дополнительный параметр TCP_NODELAY. Может использоваться только для сокета TCP.	BOOL	ИСТИНА*1: дополнительный параметр TCP_NODELAY включен ЛОЖЬ: дополнительный параметр TCP_NODELAY отключен	ЛОЖЬ

*1. При установке значения ИСТИНА отключается алгоритм Нейгла. В этом случае даже данные очень небольшого объема не передаются совместно.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Эту команду можно использовать после открытия дескриптора сокета с помощью команды SktTCPAccept или SktTCPConnect, но до запуска передачи данных командой SktTCPRcv, SktTCPSend или SktClearBuf. Если эта команда будет выполнена после запуска передачи данных, произойдет ошибка.
- Дополнительный параметр сокета должен быть настроен для каждого дескриптора, указанного в *Socket*. Настроенный дополнительный параметр сокета действует, когда дескриптор открыт. После закрытия дескриптора командой SktClose необходимо вновь выполнить команды SktTCPAccept и SktTCPConnect, чтобы открыть дескриптор, а затем выполнить эту команду для настройки дополнительного параметра сокета.

- Допускается одновременное выполнение не более 32 экземпляров следующих команд (исключение составляют модули ЦПУ NX102): SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
  В случае модулей ЦПУ NX102 может быть выполнено максимум 64 команды.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение члена структуры в *Socket* находится за пределами допустимого диапазона.
  b) Тип данных, указанный для *OptionParam*, не поддерживается переменной *OptionType*.
  c) Сокет с указанным дескриптором уже запустил передачу данных.
  d) Тип указанного сокета не поддерживается типом дескриптора. Это происходит, например, при использовании параметра TCP_NODELAY для сокета UDP.
  e) Дескриптора, указанного параметром *Socket.Handle*, не существует.

Пример программы

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	DoTCP	BOOL	ЛОЖЬ	Обработка
	Stage	INT	0	Смена состояния
	WkSocket	_sSOCKET	(Handle:=0, SrcAdr:=(PortNo:=0, IpAdr:=''), DstAdr:=(PortNo:=0, IpAdr:=''))	Сокет
	SendSocketDat	ARRAY[0..1999] OF BYTE		Передаваемые данные
	Nodelay	BOOL	ИСТИНА	Настройка NoDelay
	SktTCPConnect_instance	SktTCPConnect		
	SktSetOption_instance	SktSetOption		
	SktTCPSend_instance	SktTCPSend		
	SktClose_instance	SktClose		

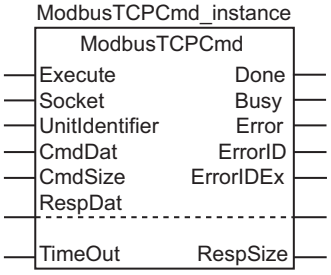
```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ((Trigger=TRUE) AND (DoTCP=FALSE) AND (_EIP_EtnOnlineSta=TRUE)) THEN
          DoTCP:=TRUE;
          Nodelay:=TRUE;
          Stage:=INT#1;
          SktTCPConnect_instance(Execute:=FALSE);// Инициализация экземпляра.
          SktSetOption_instance( // Инициализация экземпляра.
                 Execute:=FALSE,
                 OptionType:=_TCP_NODELAY,
                 OptionParam:= Nodelay);
          SktTCPSend_instance(// Инициализация экземпляра.
                 Execute:=FALSE,
```

```
                SendDat:=SendSocketDat[0]);// Формальная запись
        SktClose_instance(Execute:=FALSE);// Инициализация экземпляра.
END_IF;

IF (DoTCP=TRUE) THEN
        CASE Stage OF
        1 :// Запрос установления соединения.
                SktTCPConnect_instance(
                        Execute:=TRUE,
                        SrcTcpPort:=UINT#0,// Номер локального порта UDP: Автоматич
ески
                        DstAdr:='192.168.250.2',// Удаленный IP-адрес
                        DstTcpPort:=UINT#6000,// Номер адресуемого порта TCP
                        Socket =>WkSocket);// Сокет
                IF (SktTCPConnect_instance.Done=TRUE) THEN
                        Stage:=INT#2;// Нормальное завершение
                ELSIF (SktTCPConnect_instance.Error=TRUE) THEN
                        Stage:=INT#10; // Завершение с ошибкой
                END_IF;

        2 :// Настройка дополнительного параметра сокета
                SktSetOption_instance(
                        Execute:=TRUE,
                        Socket:=WkSocket);// Сокет
                        OptionType:=_TCP_NODELAY, // Тип дополнительного параметра
                        OptionParam:= Nodelay); // NODELAY включен
                IF (SktSetOption_instance.Done=TRUE) THEN
                        Stage:=INT#3;// Нормальное завершение
                ELSIF (SktSetOption_instance.Error=TRUE) THEN
                        Stage:=INT#20; // Завершение с ошибкой
                END_IF;

        3 :// Запрос на передачу
                SktTCPSend_instance(
                        Execute:=TRUE,
                        Socket:=WkSocket);// Сокет
                        SendDat:=SendSocketDat[0],// Передаваемые данные
                        Size:=UINT#2000);// Объем передаваемых данных
                IF (SktTCPSend_instance.Done=TRUE) THEN
                        Stage:=INT#4;// Нормальное завершение
                ELSIF (SktTCPSend_instance.Error=TRUE) THEN
                        Stage:=INT#30; // Завершение с ошибкой
                END_IF;

        4 :// Запрос на закрытие.
                SktClose_instance(
                        Execute:=TRUE,
                        Socket:=WkSocket);// Сокет
                IF (SktClose_instance.Done=TRUE) THEN
                        Stage:=INT#0;// Нормальное завершение
                ELSIF (SktClose_instance.Error=TRUE) THEN
```

```
                    Stage:= INT#40; // Завершение с ошибкой
                END_IF;

        0 :// Нормальное завершение
                DoTCP:=FALSE;
                Trigger:=FALSE;

        ELSE // Прервано по ошибке.
                DoTCP:=FALSE;
                Trigger:=FALSE;
    END_CASE;
END_IF;
```

ModbusTCPCmd

Команда ModbusTCPCmd передает команды общего назначения с использованием протокола Modbus-TCP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ModbusTCPCmd	Передача команды общего назначения по протоколу Modbus TCP	FB		ModbusTCPCmd_instance(Execute, Socket, UnitIdentifier, CmdDat, CmdSize, RespDat, TimeOut, Done, Busy, Error, ErrorID, ErrorIDEx, RespSize);

Информация о версии

Для использования этой команды требуется модуль ЦПУ NX102.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---
UnitIdentifier	Идентификатор модуля		Идентификатор модуля*1	Зависит от типа данных.	---	255
CmdDat	Данные команды		Данные команды	Зависит от типа данных.	---	---
CmdSize	Объем данных команды		Объем данных команды	1…253	Байты	1
TimeOut	Время ожидания		Время задается в масштабе 1 = 0,1 с. Если указано значение *0*, время ожидания принимается равным 2 с.	Зависит от типа данных.	0,1 с	20
RespDat	Данные ответа	Вход-выход	Данные ответа	Зависит от типа данных.	---	---
RespSize	Размер ответа	Выход	Объем данных ответа	1…253	---	---

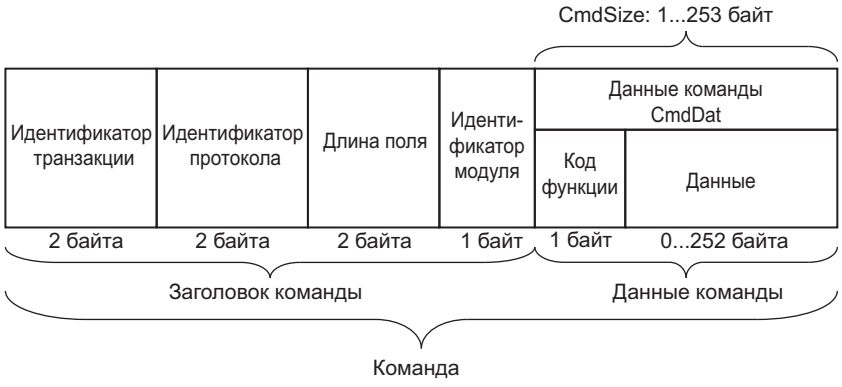
*1. При передаче команд ведомым устройствам Modbus-TCP для работы используется значение по умолчанию.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Тип данных переменной Socket* на стр. 2-1279.																			
UnitIdentifier						ОК														
CmdDat[] (массив)		ОК																		
CmdSize							ОК													
TimeOut							ОК													
RespDat[] (массив)		ОК																		
RespSize							ОК													

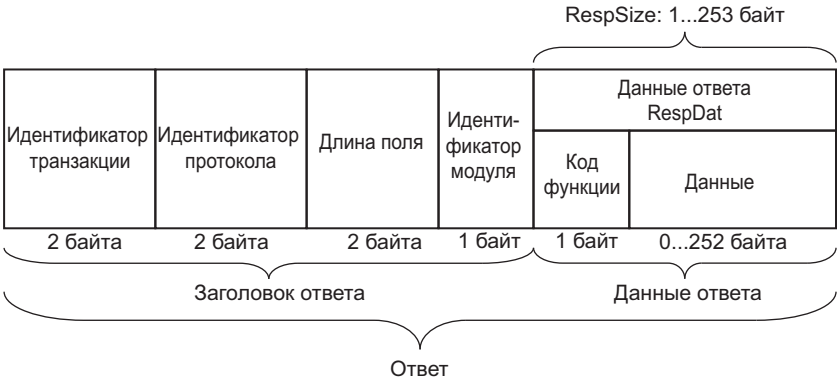
Функция

Команда ModbusTCPCmd использует протокол Modbus TCP для передачи команд общего назначения адресуемому сокету, соединение с которым устанавливается заранее путем выполнения команды SktTCPConnect.
Команда ModbusTCPCmd завершается нормально, если поступает нормальный ответ на переданную команду.

В качестве данных команды передается содержимое входной переменной CmdDat[], а объем (длина) передаваемых данных задается входной переменной *CmdSize*.



После передачи команд общего назначения команда ModbusTCPCmd записывает во входную-выходную переменную RespDat[] данные ответа, полученные от адресата.



В переменную *RespSize* выводится объем данных ответа (в байтах), содержащихся в полученном ответе.
В случае получения ответа с ошибкой значения переменных RespDat[] и *RespSize* не изменяются.

Во входной переменной *TimeOut* задается время ожидания в масштабе 1 = 100 мс. Если ответ не возвращается в течение заданного времени ожидания, возникает ошибка таймаута.

Тип данных переменной *Socket*

Для переменной *Socket* используется структурный тип данных _sSOCKET. Описание приведено в таблице ниже.

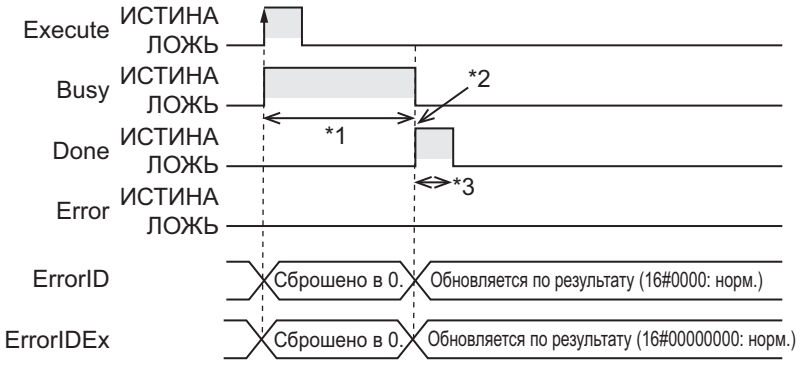
Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Сокет	_sSOCKET	---	---	---
Handle	Дескриптор	Дескриптор сокета для передачи и приема данных	UDINT	Зависит от типа данных.	---	0
SrcAdr*1	Локальный адрес	Локальный IP-адрес и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65 535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		‘’
DstAdr*1	Адрес назначения	IP-адрес назначения и номер порта	_sSOCKET_ADDRESS	---	---	---
PortNo*1	Номер порта	Номер порта	UINT	1…65 535	---	0
IpAdr*1	IP-адрес	IP-адрес или имя хоста. Для использования имени хоста должны быть настроены параметры DNS или хостов.	STRING	Зависит от типа данных.		‘’

*1. Эти члены структуры для данной команды не используются.

Временные диаграммы

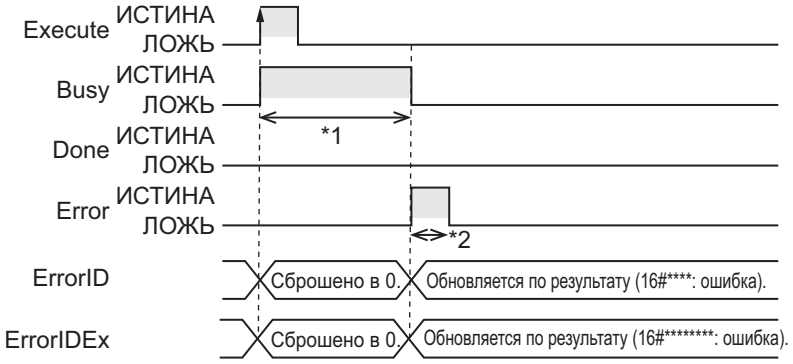
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Взаимодействие с ведомым устройством Modbus
*2. Получен ответ на переданную команду.
*3. Период выполнения задачи

● Завершение с ошибкой



*1. Взаимодействие с ведомым устройством Modbus
*2. Период выполнения задачи

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

- Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.
- Если эта команда выполняется в средстве моделирования (Simulator), то при переходе входа *Execute* из состояния ЛОЖЬ в состояние ИСТИНА выход *Done* немедленно переходит в состояние ИСТИНА. По физическому интерфейсу связи данные не передаются. Значение RespDat[] при этом не изменяется, в то время как значение *RespSize* меняется на 0.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. Если выполнение команды завершается нормально, значение *Done* меняется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временные диаграммы для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- Эту команду можно использовать только для встроенного порта EtherNet/IP модуля ЦПУ серии NX102.
- Чтобы адресуемому узлу можно было передать команду общего назначения с помощью этой команды, необходимо заранее установить соединение с этим узлом с помощью команды SktTCPConnect. Полученное при установлении соединения значение сокета следует ввести в переменную *Socket* этой команды. При этом следует указать номер порта на стороне ведомого устройства Modbus TCP (номер порта по умолчанию установлен равным 502).
- Эта команда не очищает буфер приема для сокета TCP. Если требуется очистить буфер, следует выполнить команду SktClearBuf.
- Если нужно настроить дополнительный параметр сокета, следует выполнить команду SktSetOption.
- Допускается одновременное выполнение не более 64 экземпляров следующих команд: SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
- При возникновении ошибки состояние выхода *Error* меняется на ИСТИНА. В следующей таблице приведены возможные значения переменных *ErrorID* и *ErrorIDEx* и поясняется их значение.

Значение *ErrorID*	Значение *ErrorIDEx*	Ошибка
16#0400	16#00000000	Значение *CmdSize* находится за пределами допустимого диапазона.
16#0406	16#00000000	Значение *CmdSize* привело к выходу за область массива CmdDat[].
16#0407	16#00000000	Количество принятых байтов данных превысило объем области принимаемых данных.
16#0C10	16#000000XX	По протоколу Modbus был получен ответ с кодом исключения. Код исключения указывается в позиции XX в значении 000000xx переменной ErrorIDEx. Подробные сведения о кодах исключения см. в документации по протоколу Modbus.

Значение *ErrorID*	Значение *ErrorIDEx*	Ошибка
16#0C11	16#00000000	Получены неверные данные ответа Modbus. • Неверный код функции (FuctionCode). • Неверный объем принятых данных.
16#2003	16#00000000	• В данный момент выполняется операция с сокетом. • Сокет закрыт.
16#2006	16#00000000	В течение заданного времени ожидания от адресуемого узла не был получен ответ.
16#2007	16#00000000	Значение дескриптора находится вне допустимого диапазона.
16#2008	16#00000000	Было выполнено более 64 следующих команд одновременно: SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.

Пример программы

Ниже представлен пример программы для модуля ЦПУ NX102 с IP-адресом 192.168.250.1. Когда входная переменная *Trigger* переходит в состояние ИСТИНА, программа очищает буфер, после чего передает команду Modbus адресуемому ведомому устройству (192.168.250.10, порт 502) по протоколу Modbus TCP.
Программа считывает содержимое регистра хранения по начальному адресу чтения 32 (BYTE#16#0020) в адресуемом ведомом устройстве. Для чтения значений переменных используется команда общего назначения.
В данном примере программы указывается дополнительный параметр TCP-NODELAY в соответствии с рекомендациями документа «Modbus Messaging on TCP/IP implementation guide V1.0b» («Руководство по осуществлению передачи сообщений Modbus по сети TCP/IP, версия 1.0b»).

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL		Условие выполнения
	DoModbusTrigger	BOOL		Обработка
	Nodelay	BOOL		Настройка NoDelay
	Stage	INT		Смена состояния
	Socket	_sSOCKET		Сокет
	ModbusCmdDat	ARRAY[0..4] OF BYTE		Данные команды
	ModbusDatSize	UINT		Объем данных команды
	ModbusRespDat	ARRAY[0..253] OF BYTE		Данные ответа
	ModbusRespSize	UINT		Объем данных ответа
	SktTCPConnect_instance	SktTCPConnect		
	SktSetOption_instance	SktTSetOption		
	SktClearBuf_instance	SktClearBuf		
	ModbusTCPCmd_instance	ModbusTCPCmd		
	SktClose_instance	SktClose		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EIP1_EtnOnlineSta	BOOL	☑	В сети

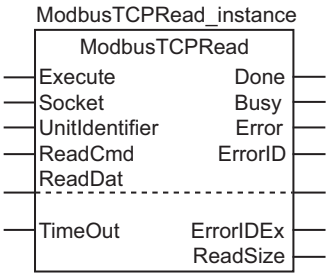
```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF (Trigger=TRUE) AND (DoModbusTrigger=FALSE) AND (_EIP1_EtnOnlineSta=TRUE) THEN
    DoModbusTrigger:=TRUE;
    Nodelay:=TRUE;

    SktTCPConnect_instance(Execute:= FALSE);
    SktSetOption_instance(
        Execute:=FALSE,
        Socket:=Socket,
        OptionType:=_eSKT_OPTION_TYPE#_TCP_NODELAY,
        OptionParam:=Nodelay);

    SktClearBuf_instance(
        Execute:=FALSE,
        Socket:=Socket);
    ModbusTCPCmd_instance(
        Execute:=FALSE,
        Socket:=Socket,
        CmdDat:=ModbusCmdDat[0],
        CmdSize:=ModbusDatSize,
        RespDat:=ModbusRespDat[0]);

    SktClose_instance(
        Execute:=FALSE,
        Socket:=Socket);

    Stage:=1; // Инициализация завершена.
END_IF;

IF (DoModbusTrigger=TRUE) THEN
    CASE Stage OF
    1:  // Запрос на соединение с сокетом
        SktTCPConnect_instance(
            Execute:=TRUE,
            SrcTcpPort:=UINT#502,
            DstAdr:='192.168.250.10',
            DstTcpPort:=UINT#502,
            Socket=>Socket);
        IF (SktTCPConnect_instance.Done=TRUE) THEN
            Stage:=2; // Соединение с сокетом завершено нормально.
        ELSIF (SktTCPConnect_instance.Error=TRUE) THEN
            Stage:=99; // Соединение с сокетом завершено с ошибкой.
        END_IF;
```

```
2:  //Запрос дополнительного параметра TCP-NODELAY
    SktSetOption_instance(
        Execute:=TRUE,
        Socket:=Socket,
        OptionType:=_eSKT_OPTION_TYPE#_TCP_NODELAY,
        OptionParam:=Nodelay);
    IF (SktSetOption_instance.Done=TRUE) THEN
        Stage:=3; // Настройка доп. параметра завершена нормально.
    ELSIF (SktSetOption_instance.Error=TRUE) THEN
        Stage:=99; // Настройка доп. параметра завершена с ошибкой.
    END_IF;

3:  // Запрос на очистку буфера
    SktClearBuf_instance(
        Execute:=TRUE,
        Socket:=Socket);
    IF (SktClearBuf_instance.Done=TRUE) THEN
        Stage:=4; // Очистка буфера завершена нормально.
    ELSIF (SktClearBuf_instance.Error=TRUE) THEN
        Stage:=99; // Очистка буфера завершена с ошибкой.
    END_IF;

4:  // Запрос на передачу команды Modbus
    ModbusCmdDat[0]:=BYTE#16#03; // Код функции (чтение переменной)
    ModbusCmdDat[1]:=BYTE#16#00; // Начальный адрес чтения (старш.)
    ModbusCmdDat[2]:=BYTE#16#20; // Начальный адрес чтения (младш.)
    ModbusCmdDat[3]:=BYTE#16#00; // Количество данных (старш.)
    ModbusCmdDat[4]:=BYTE#16#01; // Количество данных (младш.)
    ModbusDatSize:=5;
    ModbusTCPCmd_instance(
        Execute:=TRUE,
        Socket:=Socket,
        CmdDat:=ModbusCmdDat[0],
        CmdSize:=ModbusDatSize,
        RespDat:=ModbusRespDat[0],
        RespSize=>ModbusRespSize);
    IF (ModbusTCPCmd_instance.Done=TRUE) THEN
        Stage:=5; // Команда ModbusTCPCmd завершена нормально.
    ELSIF (ModbusTCPCmd_instance.Error=TRUE) THEN
        Stage:=99; // Команда ModbusTCPCmd завершена с ошибкой.
    END_IF;

5:  // Запрос на закрытие сокета
    SktClose_instance(
        Execute:=TRUE,
        Socket:=Socket);
    IF (SktClose_instance.Done=TRUE) THEN
        Stage:=6; // Закрытие сокета завершено нормально.
    ELSIF (SktClose_instance.Error=TRUE) THEN
        Stage:=99; // Закрытие сокета завершено с ошибкой.
    END_IF;
```

```
    6:  // Обработка после нормального завершения команды ModbusTCPCmd.
        Trigger:=FALSE;
        DoModbusTrigger:=FALSE;

    99:  // Обработка ошибки
        Trigger:=FALSE;
        DoModbusTrigger:=FALSE;
    END_CASE;
END_IF;
```

ModbusTCPRead

Команда ModbusTCPRead производит чтение данных, которые запрашиваются путем передачи команд чтения с использованием протокола Modbus-TCP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ModbusTCPRead	Передача команды чтения по протоколу Modbus TCP	FB		ModbusTCPRead_instance(Execute, Socket, UnitIdentifier, ReadCmd, ReadDat, TimeOut, Done, Busy, Error, ErrorID, ErrorIDEx, ReadSize);

Информация о версии

Для использования этой команды требуется модуль ЦПУ NX102.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---
UnitIdentifier	Идентификатор модуля		Идентификатор модуля*1	Зависит от типа данных.	---	255
ReadCmd	Команда чтения		Данные команды	Зависит от типа данных.	---	---
TimeOut	Время ожидания		Время ожидания задается в масштабе 1 = 0,1 с. Если указано значение *0*, время ожидания принимается равным 2 с.	Зависит от типа данных.	0,1 с	20
ReadDat	Прочитанные данные	Вход-выход	Прочитанные данные	Зависит от типа данных.	---	---
ReadSize	Размер чтения	Выход	Объем прочитанных данных	1...2000*2 1...125*3	Биты*2 Слова*3	---

*1. При передаче команд ведомым устройствам Modbus-TCP для работы используется значение по умолчанию.
*2. Этот диапазон допустимых значений применяется при чтении состояния выходов или входов (BOOL).
*3. Этот диапазон допустимых значений применяется при чтении содержимого входного регистра или регистра хранения (WORD).

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Тип данных переменной Socket* на стр. 2-1279.																			
UnitIdentifier						ОК														
ReadCmd	Подробные сведения о структуре _sMODBUS_READ см. в разделе *Тип данных переменной ReadCmd* на стр. 2-1288.																			
TimeOut							ОК													
ReadDat[] (массив)	ОК		ОК																	
ReadSize							ОК													

Функция

Команда ModbusTCPRead использует протокол Modbus TCP для передачи команд чтения адресуемому сокету, соединение с которым устанавливается заранее путем выполнения команды SktTCPConnect. Команда ModbusTCPRead сохраняет прочитанные данные и значение объема прочитанных данных в переменные *ReadDat* и *ReadSize* соответственно и завершается нормально при получении нормального ответа на переданную команду (т. е. при получении запрошенных данных).
В случае получения ответа с ошибкой значения переменных *ReadDat* и *ReadSize* не изменяются.

Во входной переменной *TimeOut* задается время ожидания в масштабе 1 = 100 мс. Если ответ не возвращается в течение заданного времени ожидания, возникает ошибка таймаута.

Диапазон допустимых значений переменной *ReadSize* (т. е. объем прочитанных данных) варьируется в зависимости от кода функции.
Каждое значение определяется объемом прочитанных данных и максимальной длиной команды.
Описание приведено в таблице ниже.

Код функции	ReadSize
_MDB_READ_COILS	1...2000 (бит)
_MDB_READ_DISCRETE_INPUTS	1...2 000 (бит)
_MDB_READ_HOLDING_REGISTERS	1...125 (слов)
_MDB_READ_INPUT_REGISTERS	1...125 (слов)

Переменная, в которую должны сохраняться прочитанные данные, указывается с помощью входной-выходной переменной *ReadDat*.
Тип данных, который можно использовать для переменной *ReadDat*, зависит от кода функции.
Описание приведено в таблице ниже.

Код функции	Тип данных
_MDB_READ_COILS	BOOL BOOL[]
_MDB_READ_DISCRETE_INPUTS	BOOL BOOL[]

Код функции	Тип данных
_MDB_READ_HOLDING_REGISTERS	WORD WORD[]
_MDB_READ_INPUT_REGISTERS	WORD WORD[]

Тип данных переменной *Socket*

См. *Тип данных переменной Socket* на стр. 2-1279 для команды ModbusTCPCmd.

Тип данных переменной *ReadCmd*

Для переменной *ReadCmd* используется структурный тип данных _sMODBUS_READ.
Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
ReadCmd	Команда чтения	Данные команды	_sMODBUS_READ	Зависит от типа данных.	---	---
Fun	Код функции	Код функции	_eMDB_FUN	_MDB_READ_COILS _MDB_READ_DISCRETE_INPUTS _MDB_READ_HOLDING_REGISTERS _MDB_READ_INPUT_REGISTERS	---	_MDB_READ_COILS
ReadAdr	Адрес для чтения	Начальный адрес чтения	UINT	Зависит от типа данных.	---	0
ReadSize	Размер чтения	Размер чтения	UINT	Зависит от кода функции.	---*1	1

*1. Используются те же единицы, что и единицы значения, указанного в *ReadCmd.Fun*.

● Тип данных переменной *FUN*

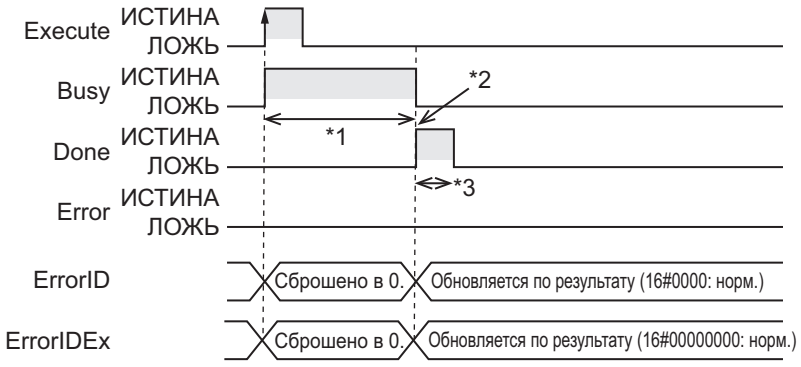
Для переменной *Fun* используется перечислимый тип данных _eMDB_FUN.
Значения перечислителей перечислимого типа _eMDB_FUN приведены в таблице ниже:

Перечислитель	Значение
_MDB_READ_COILS	Чтение выходов (бит)
_MDB_READ_DISCRETE_INPUTS	Чтение входов (бит)
_MDB_READ_HOLDING_REGISTERS	Чтение регистров хранения (слово)
_MDB_READ_INPUT_REGISTERS	Чтение входных регистров (слово)

Временные диаграммы

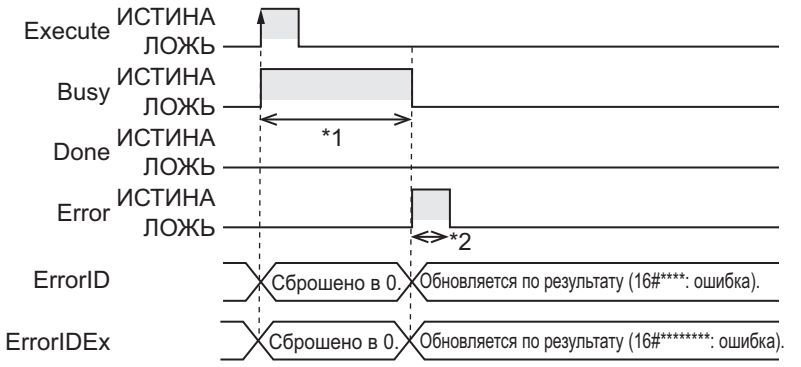
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Взаимодействие с ведомым устройством Modbus
*2. Получен ответ на переданную команду.
*3. Период выполнения задачи

● Завершение с ошибкой



*1. Взаимодействие с ведомым устройством Modbus
*2. Период выполнения задачи

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

- Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.
- Если эта команда выполняется в средстве моделирования (Simulator), то при переходе входа *Execute* из состояния ЛОЖЬ в состояние ИСТИНА выход *Done* немедленно переходит в

состояние ИСТИНА. По физическому интерфейсу связи данные не передаются. Значение ReadDat[] при этом не изменяется, в то время как значение *ReadSize* меняется на 0.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. Если выполнение команды завершается нормально, значение *Done* меняется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временные диаграммы для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- Эту команду можно использовать только для встроенного порта EtherNet/IP модуля ЦПУ серии NX102.
- Чтобы адресуемому узлу можно было передать команду общего назначения с помощью этой команды, необходимо заранее установить соединение с этим узлом с помощью команды SktTCPConnect. Полученное при установлении соединения значение сокета следует ввести в переменную *Socket* этой команды. При этом следует указать номер порта на стороне ведомого устройства Modbus TCP (номер порта по умолчанию установлен равным 502).
- Эта команда не очищает буфер приема для сокета TCP. Если требуется очистить буфер, следует выполнить команду SktClearBuf.
- Если нужно настроить дополнительный параметр сокета, следует выполнить команду SktSetOption.
- Допускается одновременное выполнение не более 64 экземпляров следующих команд: SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
- При возникновении ошибки состояние выхода *Error* меняется на ИСТИНА. В следующей таблице приведены возможные значения переменных *ErrorID* и *ErrorIDEx* и поясняется их значение.

Значение *ErrorID*	Значение *ErrorIDEx*	Ошибка
16#0400	16#00000000	Значение *ReadCmd.Fun* находится за пределами допустимого диапазона. Значение *ReadCmd.ReadSize* находится за пределами допустимого диапазона.
16#0406	16#00000000	Значение *ReadCmd.ReadSize* привело к выходу за область массива ReadDat[].
16#0407	16#00000000	Количество принятых байтов данных превысило объем области принимаемых данных.
16#0419	16#00000000	Тип данных, указанный в ReadDat[], не соответствует типу данных *ReadCmd.Fun*.
16#0C10	16#000000XX	По протоколу Modbus был получен ответ с кодом исключения. Код исключения указывается в позиции XX в значении 000000xx переменной ErrorIDEx. Подробные сведения о кодах исключения см. в документации по протоколу Modbus.
16#0C11	16#00000000	Получены неверные данные ответа Modbus. • Неверный код функции (FuctionCode). • Неверный объем принятых данных.

Значение *ErrorID*	Значение *ErrorIDEx*	Ошибка
16#2003	16#00000000	• В данный момент выполняется операция с сокетом. • Сокет закрыт.
16#2006	16#00000000	В течение заданного времени ожидания от адресуемого узла не был получен ответ.
16#2007	16#00000000	Значение дескриптора находится вне допустимого диапазона.
16#2008	16#00000000	Было выполнено более 64 следующих команд одновременно: SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.

Пример программы

Ниже представлен пример программы для модуля ЦПУ NX102 с IP-адресом 192.168.250.1.
Когда входная переменная *Trigger* переходит в состояние ИСТИНА, программа очищает буфер, после чего передает команду Modbus адресуемому ведомому устройству (192.168.250.10, порт 502) по протоколу Modbus TCP.
Программа считывает состояние выхода по начальному адресу чтения 19 в адресуемом ведомом устройстве. Для чтения значений переменных используется команда чтения.
В данном примере программы указывается дополнительный параметр TCP-NODELAY в соответствии с рекомендациями документа «Modbus Messaging on TCP/IP implementation guide V1.0b» («Руководство по осуществлению передачи сообщений Modbus по сети TCP/IP, версия 1.0b»).

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	По умолчанию	Комментарий
	Trigger	BOOL		Условие выполнения
	DoModbusTrigger	BOOL		Обработка
	Nodelay	BOOL		Настройка NoDelay
	Stage	INT		Смена состояния
	Socket	_sSOCKET		Сокет
	ModbusReadCmd	_sMODBUS_READ		Команда чтения
	ModbusRespDat	BOOL		Прочитанные данные
	ModbusReadSize	UINT		Объем прочитанных данных
	SktTCPConnect_instance	SktTCPConnect		
	SktSetOption_instance	SktTSetOption		
	SktClearBuf_instance	SktClearBuf		
	ModbusTCPRead_instance	ModbusTCPRead		
	SktClose_instance	SktClose		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EIP1_EtnOnlineSta	BOOL	☐	В сети

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF (Trigger=TRUE) AND (DoModbusTrigger=FALSE) AND (_EIP1_EtnOnlineSta=TRUE) THEN
    DoModbusTrigger:=TRUE;
    Nodelay:=TRUE;

    SktTCPConnect_instance(Execute:= FALSE);
    SktSetOption_instance(
        Execute:=FALSE,
        Socket:=Socket,
        OptionType:=_eSKT_OPTION_TYPE#_TCP_NODELAY,
        OptionParam:=Nodelay);

    SktClearBuf_instance(
        Execute:=FALSE,
        Socket:=Socket);
    ModbusTCPRead_instance(
        Execute:=FALSE,
        Socket:=Socket,
        ReadCmd:=ModbusReadCmd,
        ReadDat:=ModbusReadDat);

    SktClose_instance(
        Execute:=FALSE,
        Socket:=Socket);

    Stage:=1; // Инициализация завершена.
END_IF;

IF (DoModbusTrigger=TRUE) THEN
    CASE Stage OF
    1: // Запрос на соединение с сокетом
        SktTCPConnect_instance(
            Execute:=TRUE,
            SrcTcPPort:=UINT#502,
            DstAdr:='192.168.250.10',
            DstTcpPort:=UINT#502,
            Socket=>Socket);
        IF (SktTCPConnect_instance.Done=TRUE) THEN
            Stage:=2; // Соединение с сокетом завершено нормально.
        ELSIF (SktTCPConnect_instance.Error=TRUE) THEN
            Stage:=99; // Соединение с сокетом завершено с ошибкой.
        END_IF;

    2: //Запрос дополнительного параметра TCP-NODELAY
        SktSetOption_instance(
            Execute:=TRUE,
            Socket:=Socket,
            OptionType:=_eSKT_OPTION_TYPE#_TCP_NODELAY,
            OptionParam:=Nodelay);
```

```
    IF (SktSetOption_instance.Done=TRUE) THEN
        Stage:=3; // Настройка доп. параметра завершена нормально.
    ELSIF (SktSetOption_instance.Error=TRUE) THEN
        Stage:=99; // Настройка доп. параметра завершена с ошибкой.
    END_IF;

3: // Запрос на очистку буфера
    SktClearBuf_instance(
        Execute:=TRUE,
        Socket:=Socket);

    IF (SktClearBuf_instance.Done=TRUE) THEN
        Stage:=4; // Очистка буфера завершена нормально.
    ELSIF (SktClearBuf_instance.Error=TRUE) THEN
        Stage:=99; // Очистка буфера завершена с ошибкой.
    END_IF;

4: // Запрос чтения по протоколу Modbus
    ModbusReadCmd.Fun:=_MDB_READ_COILS; // Код функции
    ModbusReadCmd.ReadAdr:=19; // Адрес чтения
    ModbusReadCmd.ReadSize:=1; // Размер чтения

    ModbusTCPRead_instance(
        Execute:=TRUE,
        Socket:=Socket,
        ReadCmd:=ModbusReadCmd,
        ReadDat:=ModbusReadDat,
        ReadSize=>ModbusReadSize);

    IF (ModbusTCPRead_instance.Done=TRUE) THEN
        Stage:=5; // Команда ModbusTCPRead завершена нормально.
    ELSIF (ModbusTCPRead_instance.Error=TRUE) THEN
        Stage:=99; // Команда ModbusTCPRead завершена с ошибкой.
    END_IF;

5: // Запрос на закрытие сокета
    SktClose_instance(
        Execute:=TRUE,
        Socket:=Socket);
    IF (SktClose_instance.Done=TRUE) THEN
        Stage:=6; // Закрытие сокета завершено нормально.
    ELSIF (SktClose_instance.Error=TRUE) THEN
        Stage:=99; // Закрытие сокета завершено с ошибкой.
    END_IF;

6: // Обработка после нормального завершения команды ModbusTCPRead.
    Trigger:=FALSE;
    DoModbusTrigger:=FALSE;

99: // Обработка ошибки
    Trigger:=FALSE;
```

```
            DoModbusTrigger:=FALSE;
        END_CASE;
    END_IF;
```

ModbusTCPWrite

Команда ModbusTCPWrite передает команды записи с использованием протокола Modbus-TCP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ModbusTCPWrite	Передача команды записи по протоколу Modbus TCP	FB	ModbusTCPWrite_instance ModbusTCPWrite Execute, Socket, UnitIdentifier, WriteCmd, WriteDat, TimeOut Done, Busy, Error, ErrorID, ErrorIDEx	ModbusTCPWrite_instance(Execute, Socket, UnitIdentifier, WriteCmd, WriteDat, TimeOut, Done, Busy, Error, ErrorID, ErrorIDEx);

Информация о версии

Для использования этой команды требуется модуль ЦПУ NX102.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Socket	Сокет	Вход	Сокет	---	---	---
UnitIdentifier	Идентификатор модуля		Идентификатор модуля*1	Зависит от типа данных.	---	255
WriteCmd	Команда записи		Данные команды	Зависит от типа данных.	---	---
WriteDat	Записываемые данные		Записываемые данные	Зависит от типа данных.	---	---
TimeOut	Время ожидания		Время ожидания задается в масштабе 1 = 0,1 с. Если указано значение *0*, время ожидания принимается равным 2 с.	Зависит от типа данных.	0,1 с	20

*1. При передаче команд ведомым устройствам Modbus-TCP для работы используется значение по умолчанию.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Socket	Подробные сведения о структуре _sSOCKET см. в разделе *Тип данных переменной Socket* на стр. 2-1279.																			
UnitIdentifier						OK														
WriteCmd	Подробные сведения о структуре _sMODBUS_WRITE см. в разделе *Тип данных переменной WriteCmd* на стр. 2-1297.																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
WriteDat[] (массив)	ОК		ОК																	
TimeOut							ОК													

Функция

Команда ModbusTCPWrite использует протокол Modbus TCP для передачи команд записи адресуемому сокету, соединение с которым устанавливается заранее путем выполнения команды SktTCPConnect.
Команда ModbusTCPWrite завершается нормально, если поступает нормальный ответ на переданную команду.

Во входной переменной *TimeOut* задается время ожидания в масштабе 1 = 100 мс.
Если ответ не возвращается в течение заданного времени ожидания, возникает ошибка таймаута.

Диапазон допустимых значений переменной *WriteSize* (т. е. объем записываемых данных) варьируется в зависимости от кода функции.
Каждое значение определяется объемом записываемых данных и максимальной длиной команды.
Описание приведено в таблице ниже.

Код функции	WriteSize
_MDB_WRITE_SINGLE_COIL	1 (бит)
_MDB_WRITE_SINGLE_REGISTER	1 (слово)
_MDB_WRITE_MULTIPLE_COILS	1...1 968 (бит)
_MDB_WRITE_MULTIPLE_REGISTERS	1...123 (слов)

Используйте входную переменную *WriteDat* для указания данных, которые должны быть записаны.
Тип данных, который можно использовать для переменной *WriteDat*, зависит от кода функции.
Описание приведено в таблице ниже.

Код функции	Тип данных
_MDB_WRITE_SINGLE_COIL	BOOL BOOL[]
_MDB_WRITE_SINGLE_REGISTER	WORD WORD[]
_MDB_WRITE_MULTIPLE_COILS	BOOL BOOL[]
_MDB_WRITE_MULTIPLE_REGISTERS	WORD WORD[]

Тип данных переменной *Socket*

См. *Тип данных переменной Socket* на стр. 2-1279 для команды ModbusTCPCmd.

Тип данных переменной *WriteCmd*

Для переменной *WriteCmd* используется структурный тип данных _sMODBUS_WRITE.
Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
WriteCmd	Команда записи	Данные команды	_sMODBUS_WRITE	Зависит от типа данных.	---	---
Fun	Код функции	Код функции	_eMDB_FUN	_MDB_WRITE_SINGLE_COIL _MDB_WRITE_SINGLE_REGISTER _MDB_WRITE_MULTIPLE_COILS _MDB_WRITE_MULTIPLE_REGISTERS	---	_MDB_WRITE_SINGLE_COIL
WriteAdr	Адрес для записи	Начальный адрес записи	UINT	Зависит от типа данных.	---	0
WriteSize	Размер записи	Размер записи	UINT	Зависит от кода функции.	---*1	1

*1. Используются те же единицы, что и единицы значения, указанного в *WriteCmd.Fun*.

● Тип данных переменной *FUN*

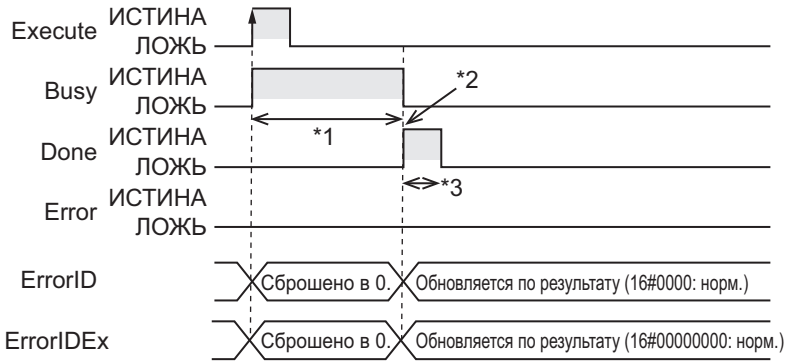
Для переменной *Fun* используется перечислимый тип данных _eMDB_FUN.
Значения перечислителей перечислимого типа _eMDB_FUN приведены в таблице ниже:

Перечислитель	Значение
_MDB_WRITE_SINGLE_COIL	Запись в выход (бит)
_MDB_WRITE_SINGLE_REGISTER	Запись в регистр хранения (слово)
_MDB_WRITE_MULTIPLE_COILS	Запись в несколько выходов (бит)
_MDB_WRITE_MULTIPLE_REGISTERS	Запись в несколько регистров хранения (слово)

Временные диаграммы

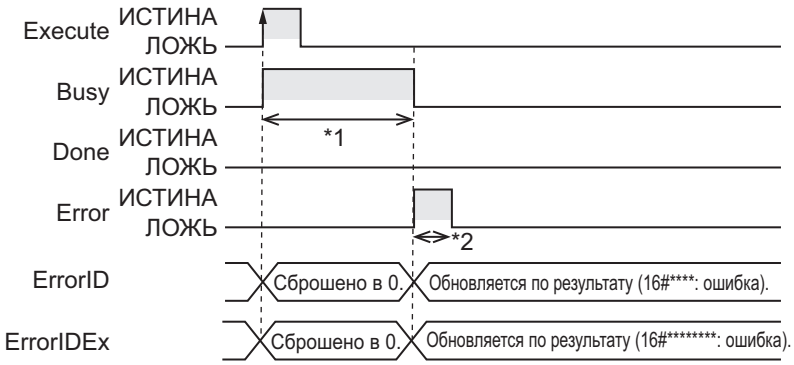
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Взаимодействие с ведомым устройством Modbus
*2. Получен ответ на переданную команду.
*3. Период выполнения задачи

● Завершение с ошибкой



*1. Взаимодействие с ведомым устройством Modbus
*2. Период выполнения задачи

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

- Дополнительные сведения о функциях, связанных со службами сокетов, см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

- Если эта команда выполняется в средстве моделирования (Simulator), то при переходе входа *Execute* из состояния ЛОЖЬ в состояние ИСТИНА выход *Done* немедленно переходит в состояние ИСТИНА. По физическому интерфейсу связи данные не передаются.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. Если выполнение команды завершается нормально, значение *Done* меняется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временные диаграммы для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- Эту команду можно использовать только для встроенного порта EtherNet/IP модуля ЦПУ серии NX102.
- Чтобы адресуемому узлу можно было передать команду общего назначения с помощью этой команды, необходимо заранее установить соединение с этим узлом с помощью команды SktTCPConnect. Полученное при установлении соединения значение сокета следует ввести в переменную *Socket* этой команды. При этом следует указать номер порта на стороне ведомого устройства Modbus TCP (номер порта по умолчанию установлен равным 502).
- Эта команда не очищает буфер приема для сокета TCP. Если требуется очистить буфер, следует выполнить команду SktClearBuf.
- Если нужно настроить дополнительный параметр сокета, следует выполнить команду SktSetOption.
- Допускается одновременное выполнение не более 64 экземпляров следующих команд: SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.
- При возникновении ошибки состояние выхода *Error* меняется на ИСТИНА. В следующей таблице приведены возможные значения переменных *ErrorID* и *ErrorIDEx* и поясняется их значение.

Значение *ErrorID*	Значение *ErrorIDEx*	Ошибка
16#0400	16#00000000	Значение *WriteCmd.Fun* находится за пределами допустимого диапазона. Значение *WriteCmd.WriteSize* находится за пределами допустимого диапазона.
16#0406	16#00000000	Значение *WriteCmd.WriteSize* привело к выходу за область массива WriteDat[].
16#0419	16#00000000	Тип данных, указанный в WriteDat[], не соответствует типу данных *WriteCmd.Fun*.
16#0C10	16#000000XX	По протоколу Modbus был получен ответ с кодом исключения. Код исключения указывается в позиции XX в значении 000000xx переменной ErrorIDEx. Подробные сведения о кодах исключения см. в документации по протоколу Modbus.
16#0C11	16#00000000	Получены неверные данные ответа Modbus. • Неверный код функции (FuctionCode). • Неверный объем принятых данных.
16#2003	16#00000000	• В данный момент выполняется операция с сокетом. • Сокет закрыт.

Значение *ErrorID*	Значение *ErrorIDEx*	Ошибка
16#2006	16#00000000	В течение заданного времени ожидания от адресуемого узла не был получен ответ.
16#2007	16#00000000	Значение дескриптора находится вне допустимого диапазона.
16#2008	16#00000000	Было выполнено более 64 следующих команд одновременно: SktUDPCreate, SktUDPRcv, SktUDPSend, SktTCPAccept, SktTCPConnect, SktTCPRcv, SktTCPSend, SktGetTCPStatus, SktClose, SktClearBuf, SktSetOption, ModbusTCPCmd, ModbusTCPRead и ModbusTCPWrite.

Пример программы

Ниже представлен пример программы для модуля ЦПУ NX102 с IP-адресом 192.168.250.1.
Когда входная переменная *Trigger* переходит в состояние ИСТИНА, программа очищает буфер, после чего передает команду Modbus адресуемому ведомому устройству (192.168.250.10, порт 502) по протоколу Modbus TCP.
Включается выход по начальному адресу записи 149 в адресуемом ведомом устройстве. Для записи значений переменных используется команда записи.
В данном примере программы указывается дополнительный параметр TCP-NODELAY в соответствии с рекомендациями документа «Modbus Messaging on TCP/IP implementation guide V1.0b» («Руководство по осуществлению передачи сообщений Modbus по сети TCP/IP, версия 1.0b»).

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	По умолчанию	Комментарий
	Trigger	BOOL		Условие выполнения
	DoModbusTrigger	BOOL		Обработка
	Nodelay	BOOL		Настройка NoDelay
	Stage	INT		Смена состояния
	Socket	_sSOCKET		Сокет
	ModbusWriteCmd	_sMODBUS_WRITE		Команда записи
	ModbusWriteDat	BOOL		Записываемые данные
	SktTCPConnect_instance	SktTCPConnect		
	SktSetOption_instance	SktTSetOption		
	SktClearBuf_instance	SktClearBuf		
	ModbusTCPWrite_instance	ModbusTCPWrite		
	SktClose_instance	SktClose		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_EIP1_EtnOnlineSta	BOOL	☑	В сети

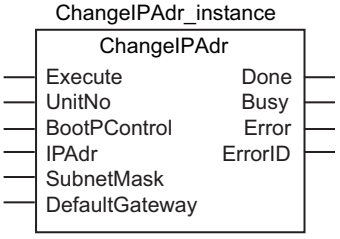
```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF (Trigger=TRUE) AND (DoModbusTrigger=FALSE) AND (_EIP1_EtnOnlineSta=TRUE) THEN
    DoModbusTrigger:=TRUE;
```

```
    Nodelay:=TRUE;

    SktTCPConnect_instance(Execute:= FALSE);
    SktSetOption_instance(
        Execute:=FALSE,
        Socket:=Socket,
        OptionType:=_eSKT_OPTION_TYPE#_TCP_NODELAY,
        OptionParam:=Nodelay);

    SktClearBuf_instance(
        Execute:=FALSE,
        Socket:=Socket);
    ModbusTCPWrite_instance(
        Execute:=FALSE,
        Socket:=Socket,
        WriteCmd:=ModbusWriteCmd,
        WriteDat:=ModbusWriteDat);

    SktClose_instance(
        Execute:=FALSE,
        Socket:=Socket);

    Stage:=1; // Инициализация завершена.
END_IF;

IF (DoModbusTrigger=TRUE) THEN
    CASE Stage OF
    1: // Запрос на соединение с сокетом
        SktTCPConnect_instance(
            Execute:=TRUE,
            SrcTcpPort:=UINT#502,
            DstAdr:='192.168.250.10',
            DstTcpPort:=UINT#502,
            Socket=>Socket);
        IF (SktTCPConnect_instance.Done=TRUE) THEN
            Stage:=2; // Соединение с сокетом завершено нормально.
        ELSIF (SktTCPConnect_instance.Error=TRUE) THEN
            Stage:=99; // Соединение с сокетом завершено с ошибкой.
        END_IF;

    2: // Запрос дополнительного параметра TCP-NODELAY
        SktSetOption_instance(
            Execute:=TRUE,
            Socket:=Socket,
            OptionType:=_eSKT_OPTION_TYPE#_TCP_NODELAY,
            OptionParam:=Nodelay);
        IF (SktSetOption_instance.Done=TRUE) THEN
            Stage:=3; // Настройка доп. параметра завершена нормально.
        ELSIF (SktSetOption_instance.Error=TRUE) THEN
            Stage:=99; // Настройка доп. параметра завершена с ошибкой.
        END_IF;
```

```
    3: // Запрос на очистку буфера
        SktClearBuf_instance(
            Execute:=TRUE,
            Socket:=Socket);
        IF (SktClearBuf_instance.Done=TRUE) THEN
            Stage:=4; // Очистка буфера завершена нормально.
        ELSIF (SktClearBuf_instance.Error=TRUE) THEN
            Stage:=99; // Очистка буфера завершена с ошибкой.
        END_IF;

    4: // Запрос записи по протоколу Modbus
        ModbusWriteCmd.Fun:=_MDB_WRITE_SINGLE_COIL; // Код функции
        ModbusWriteCmd.WriteAdr:=149; // Адрес записи
        ModbusWriteCmd.WriteSize:=1; // Размер записи

        ModbusTCPWrite_instance(
            Execute:=TRUE,
            Socket:=Socket,
            WriteCmd:=ModbusWriteCmd,
            WriteDat:=ModbusWriteDat);

        IF (ModbusTCPWrite_instance.Done=TRUE) THEN
            Stage:=5; // Команда ModbusTCPWrite завершена нормально.
        ELSIF (ModbusTCPWrite_instance.Error=TRUE) THEN
            Stage:=99; // Команда ModbusTCPWrite завершена с ошибкой.
        END_IF;

    5: // Запрос на закрытие сокета
        SktClose_instance(
            Execute:=TRUE,
            Socket:=Socket);
        IF (SktClose_instance.Done=TRUE) THEN
            Stage:=6; // Закрытие сокета завершено нормально.
        ELSIF (SktClose_instance.Error=TRUE) THEN
            Stage:=99; // Закрытие сокета завершено с ошибкой.
        END_IF;

    6: // Обработка после нормального завершения команды ModbusTCPWrite.
        Trigger:=FALSE;
        DoModbusTrigger:=FALSE;

    99: // Обработка ошибки
        Trigger:=FALSE;
        DoModbusTrigger:=FALSE;
    END_CASE;
END_IF;
```

ChangeIPAdr

Команда ChangeIPAdr изменяет IP-адрес встроенного порта EtherNet/IP модуля ЦПУ или IP-адрес модуля интерфейса EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ChangeIPAdr	Изменение IP-адреса	FB		ChangeIPAdr_instance(Execute, UnitNo, BootPControl, IPAdr, SubnetMask, DefaultGateway, Done, Busy, Error, ErrorID);

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitNo	Номер модуля	Вход	Номер модуля, для которого нужно изменить IP-адрес	_CBU_CPU *1, _CBU_CPU_Port1 *2, _CBU_CPU_Port2 *3, _CBU_No00..._CBU_No15 *4	---	_CBU_No00
BootPControl	Способ назначения IP-адреса и время установки значения		Способ получения IP-адреса и время установки значения	0...3 *5		0
IPAdr[] (массив)*6	IP-адрес		IP-адрес	*7		---
Subnet Mask[] (массив)*6	Маска подсети		Маска подсети			
Default Gateway[] (массив)*6	Шлюз по умолчанию		Шлюз по умолчанию			

*1. Может быть указано для модуля ЦПУ серии NJ.
*2. Может быть указано для порта 1 модуля ЦПУ серии NX. Вместо _CBU_CPU_Port1 можно указать _CBU_CPU.
*3. Может быть указано для порта 2 модуля ЦПУ серии NX.
*4. Может быть указано для модуля ЦПУ серии NJ.
*5. Возможные значения для порта 1 модуля ЦПУ серии NX и для модуля ЦПУ серии NJ: *0...2*. Возможные значения для порта 2 модуля ЦПУ серии NX: *0...3*.
*6. Это массив с четырьмя элементами с номерами от 0 до 3.
*7. Допустимый диапазон зависит от того, что указано в переменной *UnitNo* (номер модуля): встроенный порт EtherNet/IP или модуль интерфейса EtherNet/IP.

Дополнительные сведения см. в разделе *Функция* на стр. 2-1304 ниже.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitNo	Сведения о перечислителях перечислимого типа _eUnitNo см. в разделе *Функция* на стр. 2-1304.																			
BootPControl							ОК													
IPAdr[] (массив)		ОК																		
	Следует указать массив.																			
Subnet Mask[] (массив)		ОК																		
	Следует указать массив.																			
Default Gateway[] (массив)		ОК																		
	Следует указать массив.																			

Функция

Команда ChangeIPAdr изменяет IP-адрес встроенного порта EtherNet/IP или модуля интерфейса EtherNet/IP, который указан с помощью параметра *UnitNo* (номер модуля), в соответствии со способом назначения IP-адреса и временем его установки, которые выбраны с помощью параметра *BootPControl*.

Если в параметре *UnitNo* указан встроенный порт EtherNet/IP, порт переходит в состояние «Link OFF» («Отсутствие соединения») по окончании выполнения команды, а затем переходит в состояние «Link ON» («Соединение установлено») с новым IP-адресом.
Если в параметре *UnitNo* указан модуль интерфейса EtherNet/IP, модуль интерфейса EtherNet/IP перезапускается по окончании выполнения команды. Связь с использованием нового IP-адреса становится возможной по завершении перезапуска модуля.

Эту команду можно использовать для изменения IP-адреса встроенного порта EtherNet/IP или модуля интерфейса EtherNet/IP с терминала HMI.

Для параметра *UnitNo* используется перечислимый тип данных _eUnitNo. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_CBU_CPU*1	Встроенный порт EtherNet/IP
_CBU_CPU_Port1*2	Встроенный порт связи EtherNet/IP 1
_CBU_CPU_Port2*3	Встроенный порт связи EtherNet/IP 2
_CBU_No00..._CBU_No15 *4	Номер модуля 00...15 модуля EtherNet/IP

*1. Может быть указано для модуля ЦПУ серии NJ.
*2. Может быть указано для модуля ЦПУ серии NX. Вместо _CBU_CPU_Port1 можно указать _CBU_CPU.
*3. Может быть указано для модуля ЦПУ серии NX. Это значение нельзя использовать для модулей ЦПУ, не имеющих порт связи 2.
*4. Может быть указано для модуля ЦПУ серии NJ.

Значение параметра *BootPControl* определяет способ получения нового IP-адреса и время его установки, что показано в таблице ниже.
Возможные значения переменной *BootPControl* для порта 1 модуля ЦПУ серии NX и для модуля ЦПУ серии NJ: *0...2*.
Возможные значения для порта 2 модуля ЦПУ серии NX: *0...3*.

Значение *BootPControl*	Способ получения IP-адреса	Время изменения IP-адреса
0	Значение IP-адреса определяется параметрами IPAdr[] (IP-адрес), SubnetMask[] (маска подсети) и DefaultGateway[] (шлюз по умолчанию).	IP-адрес устанавливается только один раз при каждом выполнении команды (фиксированная настройка).
1	IP-адрес поступает от сервера BOOTP.	IP-адрес устанавливается один раз при выполнении команды, а затем один раз при каждом включении питания контроллера.
2	IP-адрес поступает от сервера BOOTP.	IP-адрес устанавливается только один раз при каждом выполнении команды (фиксированная настройка).
3	Порт настроен как неиспользуемый порт. Любой существующий IP-адрес удаляется.	IP-адрес устанавливается только один раз при каждом выполнении команды (фиксированная настройка).

Значения IP-адреса, маски подсети и шлюза по умолчанию задаются с помощью элементов 0...3 массивов IPAdr[], SubnetMask[] и DefaultGateway[] соответственно. Например, если новый IP-адрес = 130.58.17.32, в элементы массива IPAdr[] нужно записать следующие значения: IPAdr[0] = BYTE#16#82, IPAdr[1] = BYTE#16#3A, IPAdr[2] = BYTE#16#11 и IPAdr[3] = BYTE#16#20.

Диапазоны допустимых значений для IPAdr[], SubnetMask[] и DefaultGateway[] зависят от того, что указано в переменной *UnitNo*: встроенный порт EtherNet/IP или модуль интерфейса EtherNet/IP (см. таблицу ниже). Диапазоны допустимых значений действительны, только если для переменной *BootPControl* установлено значение *0*.

Настройка *UnitNo*	Входная переменная	Диапазон допустимых значений
Встроенный порт EtherNet/IP	IPAdr[] (массив)	Перечисленные ниже IP-адреса использовать нельзя. Все остальные IP-адреса действительны. • IP-адреса, начинающиеся со 127, 0 или 255. • IP-адреса с идентификатором хоста, в котором все биты содержат только 0 или только 1 • IP-адреса класса D (224.0.0.0...239.255.255.255) • IP-адреса класса E (240.0.0.0...255.255.255.255) • IP-адреса, зарезервированные для функции AutoIP[*1] (169.254.0.0...169.254.255.255) • IP-адреса портов USB (192.168.255.0...192.168.255.255)[*2]
	Subnet Mask[] (массив)	192.0.0.0...255.255.255.252
	DefaultGateway[] (массив)	Перечисленные ниже IP-адреса использовать нельзя. Все остальные IP-адреса действительны. • IP-адреса, начинающиеся со 127, 0 или 255. • Есть только один адрес, в котором все биты содержат только 1 • IP-адреса класса D (224.0.0.0...239.255.255.255) • IP-адреса класса E (240.0.0.0...255.255.255.255) • IP-адреса, зарезервированные для функции AutoIP[*1] (169.254.0.0...169.254.255.255) • IP-адреса портов USB (192.168.255.0...192.168.255.255)[*2]

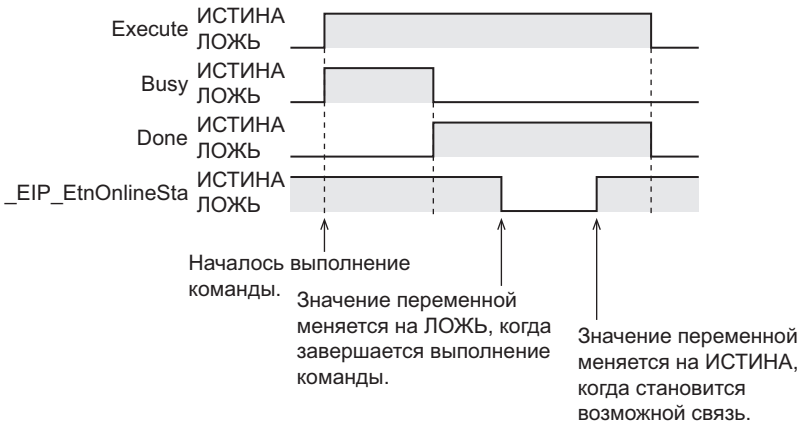
Настройка *UnitNo*	Входная переменная	Диапазон допустимых значений
Модуль интерфейса Ethernet/IP	IPAdr[] (массив)	Перечисленные ниже IP-адреса использовать нельзя. Все остальные IP-адреса действительны. • IP-адреса, начинающиеся со 127. • IP-адреса класса D (224.0.0.0...239.255.255.255) • IP-адреса класса E (240.0.0.0...255.255.255.255)
	SubnetMask[] (массив)	• 0.0.0.0 • 192.0.0.0...255.255.255.252
	DefaultGateway[] (массив)	Перечисленные ниже IP-адреса использовать нельзя. Все остальные IP-адреса действительны. • IP-адреса, начинающиеся со 127. • IP-адреса класса D (224.0.0.0...239.255.255.255) • IP-адреса класса E (240.0.0.0...255.255.255.255)

*1. AutoIP — это функция автоматического назначения IP-адреса, предусмотренная в операционной системе Windows 98 и более поздних версиях этой ОС.
*2. В модулях ЦПУ NX102 и NX1P2 порт USB отсутствует.

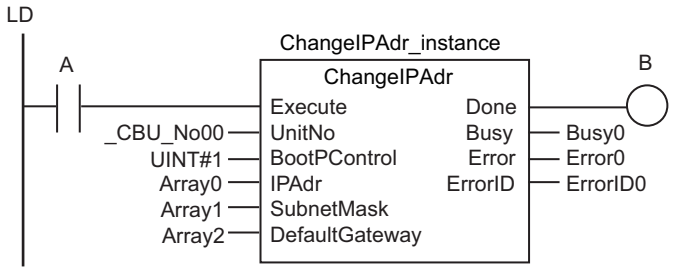
Если параметр *BootPControl* содержит значение 1 или 2, значения IPAdr[], SubnetMask[] и DefaultGateway[] игнорируются. Поэтому в этом случае значения IPAdr[], SubnetMask[] и DefaultGateway[] могут находиться за пределами допустимых диапазонов.

Если в параметре *UnitNo* указан встроенный порт EtherNet/IP, с помощью системных переменных *_EIP_EtnOnlineSta*, *_EIP1_EtnOnlineSta* и *_EIP2_EtnOnlineSta* можно проверять, возможен ли обмен данными.
Здесь в качестве примера используется системная переменная *_EIP_EtnOnlineSta*, но эта информация также относится и к системным переменным *_EIP1_EtnOnlineSta* и *_EIP2_EtnOnlineSta*.

Когда значение *Busy* меняется на ЛОЖЬ, значение *_EIP_EtnOnlineSta* также меняется на ЛОЖЬ. Когда становится возможной связь с использованием нового IP-адреса, значение *_EIP_EtnOnlineSta* меняется на ИСТИНА.



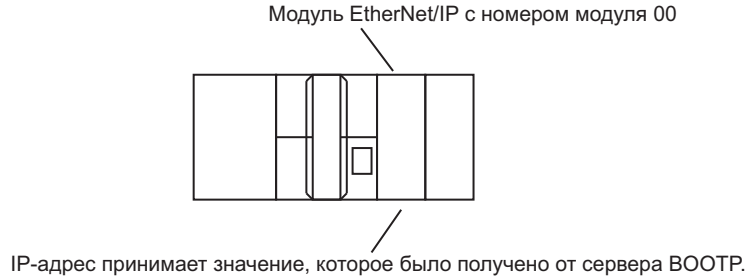
Ниже показан пример использования команды для случая, когда каждый раз при выполнении команды IP-адрес модуля EtherNet/IP с номером модуля 00 меняется на IP-адрес, получаемый от сервера BOOTP.
Когда значение переменной A (*Execute*) меняется на ИСТИНА с терминала HMI или другого устройства, IP-адрес меняется на IP-адрес, полученный от сервера BOOTP. После этого каждый раз, когда включается питание, IP-адрес меняется на IP-адрес, полученный от сервера BOOTP.



ST

```
ChangeIPAdr_instance(A,_CBU_No00,UINT#1,Array0,Array1,Array2,B,Busy0,Error0,ErrorID0);
```

Для модуля EtherNet/IP с номером модуля 00 устанавливается IP-адрес, полученный от сервера BOOTP.
После этого каждый раз, когда включается питание, IP-адрес сбрасывается к значению, полученному от сервера BOOTP.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

- Если в параметре *UnitNo* указан встроенный порт EtherNet/IP, то при выполнении команды в журнале событий регистрируются указанные ниже события.
  a) Обнаружено отсутствие соединения (Link OFF)
  b) Фиксированный IP-адрес
- Изменение IP-адреса с помощью этой команды возможно, даже если модуль ЦПУ защищен от записи.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если в параметре *UnitNo* указан встроенный порт EtherNet/IP, то по окончании выполнения команды обмен данными через встроенный порт EtherNet/IP будет временно приостановлен. Убедитесь в том, что переход встроенного порта EtherNet/IP в состояние «Link OFF» («Отсутствие соединения») не повлияет отрицательно на работу системы и безопасен.
- Если в параметре *UnitNo* указан модуль интерфейса EtherNet/IP, модуль интерфейса EtherNet/IP перезапускается по окончании выполнения команды. Убедитесь в том, что перезапуск модуля интерфейса EtherNet/IP не повлияет отрицательно на работу системы и безопасен.
- Эту команду невозможно использовать в событийной задаче. В этом случае произойдет ошибка компиляции.
- При возникновении ошибки состояние выхода *Error* меняется на ИСТИНА. В следующей таблице приведены возможные значения переменной *ErrorID* и поясняется их значение.

Значение *ErrorID*	Наименование ошибки	Описание
16#0400	Входное значение вне диапазона	Значение входной переменной находится за пределами допустимого диапазона.
16#2400	Нет права на выполнение	Команда была выполнена, когда изменение состояния было невозможно: • Когда уже выполнялась другая операция изменения параметров. • Когда производился перезапуск встроенного порта EtherNet/IP. • Когда выполнялась загрузка параметров теговых логических связей из Network Configurator.
16#2402	Слишком много одновременно выполняемых команд	Одновременно выполнялось слишком много команд ChangeIPAdr, ChangeFTPAccount и ChangeNTPServerAdr.
16#240D	Неверно настроен IP-адрес	Сетевой адрес указанного порта совпадает с сетевым адресом другого порта.

Пример программы

Рассмотрим пример программы для изменения значений IP-адреса встроенного парта EtherNet/IP на приведенные ниже фиксированные значения.

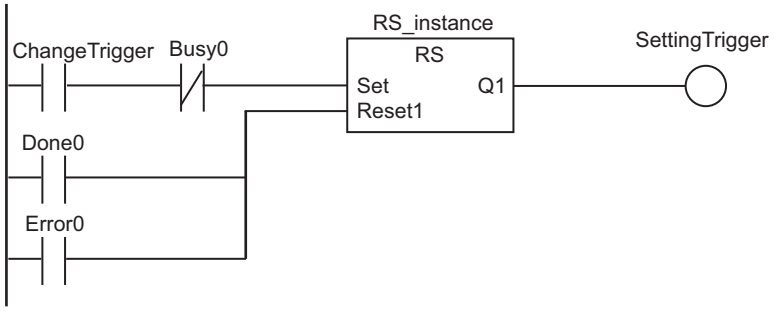
Параметр	Значение
IP-адрес	192.168.250.10
Маска подсети	255.255.255.0
Шлюз по умолчанию	0.0.0.0

Программа на языке LD

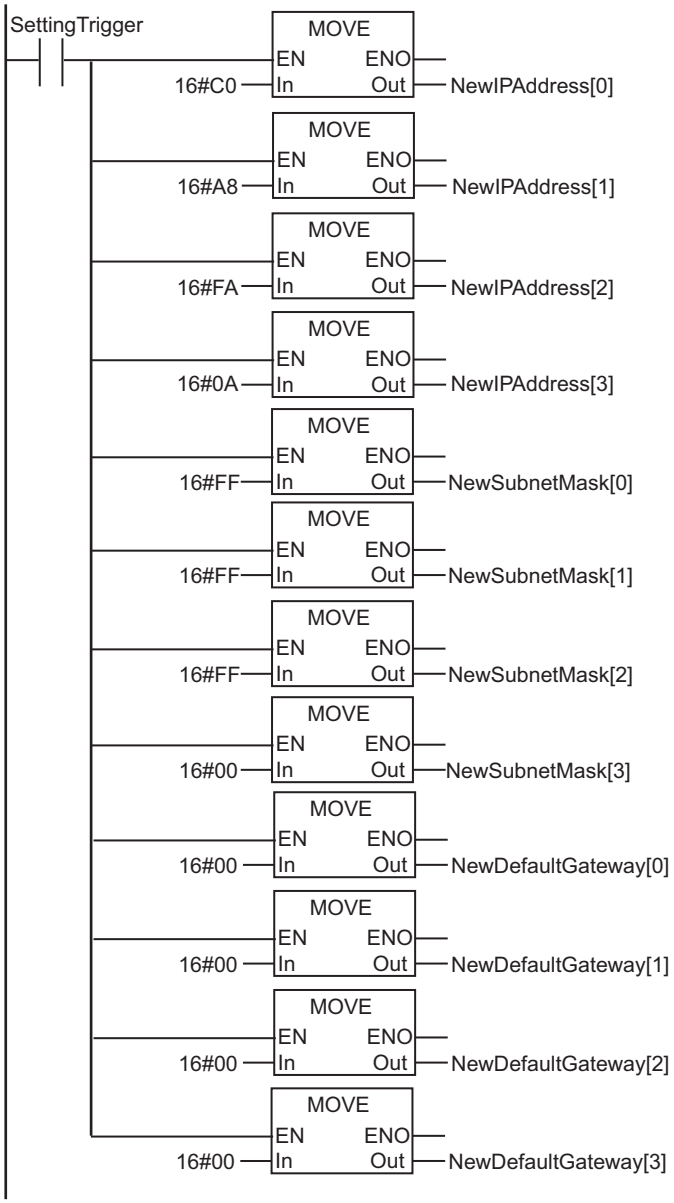
Переменная	Тип данных	Начальное значение	Комментарий
ChangeTrigger	BOOL	Ложь	Флаг изменения

Переменная	Тип данных	Начальное значение	Комментарий
SettingTrigger	BOOL	Ложь	Флаг изменения IP-адреса
Done0	BOOL	Ложь	IP-адрес изменен
Error0	BOOL	Ложь	При изменении IP-адреса произошла ошибка
Busy0	BOOL	Ложь	Изменение IP-адреса
ErrorID0	WORD	16#0	Идентификатор ошибки, произошедшей при изменении IP-адреса
NewIPAddress	ARRAY[0..3] OF BYTE	[4(16#0)]	IP-адрес
NewSubnetMask	ARRAY[0..3] OF BYTE	[4(16#0)]	Маска подсети
NewDefaultGateway	ARRAY[0..3] OF BYTE	[4(16#0)]	Шлюз по умолчанию
RS_instance	RS		
ChangeIPAdr_instance	ChangeIPAdr		

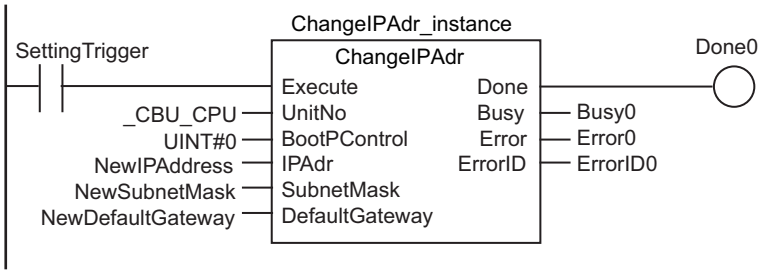
Проверка условий выполнения.



Установка IP-адреса.



Изменение IP-адреса.



Программа на языке ST

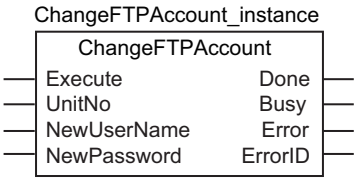
Переменная	Тип данных	Начальное значение	Комментарий
ChangeTrigger	BOOL	Ложь	Флаг изменения
SettingTrigger	BOOL	Ложь	Флаг изменения IP-адреса

Переменная	Тип данных	Начальное значение	Комментарий
Done0	BOOL	Ложь	IP-адрес изменен
Error0	BOOL	Ложь	При изменении IP-адреса произошла ошибка
Busy0	BOOL	Ложь	Изменение IP-адреса
ErrorID0	WORD	16#0	Идентификатор ошибки, произошедшей при изменении IP-адреса
NewIPAddress	ARRAY[0..3] OF BYTE	[4(16#0)]	IP-адрес
NewSubnetMask	ARRAY[0..3] OF BYTE	[4(16#0)]	Маска подсети
NewDefaultGateway	ARRAY[0..3] OF BYTE	[4(16#0)]	Шлюз по умолчанию
RS_instance	RS		
ChangeIPAdr_instance	ChangeIPAdr		

```
//Проверяем условия выполнения..
IF((ChangeTrigger=TRUE)AND(Busy0=FALSE))THEN
   SettingTrigger:= TRUE;
END_IF;

IF((Done0=TRUE)OR(Error0=TRUE))THEN
   SettingTrigger:= FALSE;
END_IF;

//Установка IP-адреса.
IF(SettingTrigger=TRUE)THEN
   NewIPAddress[0]     := 16#C0;
   NewIPAddress[1]     := 16#A8;
   NewIPAddress[2]     := 16#FA;
   NewIPAddress[3]     := 16#0A;
   NewSubnetMask[0]    := 16#FF;
   NewSubnetMask[1]    := 16#FF;
   NewSubnetMask[2]    := 16#FF;
   NewSubnetMask[3]    := 16#00;
   NewDefaultGateway[0]:= 16#00;
   NewDefaultGateway[1]:= 16#00;
   NewDefaultGateway[2]:= 16#00;
   NewDefaultGateway[3]:= 16#00;
END_IF;

//Изменение IP-адреса.
ChangeIPAdr_instance(
          Execute        := SettingTrigger,
          UnitNo         := _CBU_CPU,
          BootPControl   := UINT#0,
          IPAdr          := NewIPAddress,
          SubnetMask     := NewSubnetMask,
          DefaultGateway := NewDefaultGateway,
          Done            =>Done0,
          Busy            =>Busy0,
          Error           =>Error0,
          ErrorID         =>ErrorID0);
```

ChangeFTPAccount

Команда ChangeFTPAccount изменяет имя и пароль для входа на сервер FTP для встроенного порта EtherNet/IP модуля ЦПУ или для модуля интерфейса EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ChangeFTPAccount	Изменение учетной записи FTP	FB		ChangeFTPAccount_instance(Execute, UnitNo, NewUserName, NewPassword, Done, Busy, Error, ErrorID);

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitNo	Номер модуля	Вход	Номер модуля, для которого нужно изменить имя и пароль для входа на сервер FTP	_CBU_CPU или _CBU_No00..._CBU_No15 *1	---	_CBU_No00
NewUserName	Имя для входа		Имя для входа	От 1 до 12 однобайтовых буквенно-цифровых символов (с различением строчных и заглавных букв)		---
NewPassword	Пароль		Пароль	*2		

*1. Значение *_CBU_No00..._CBU_No15* можно задать только для модуля ЦПУ серии NJ.

*2. Допустимый диапазон зависит от того, что указано в переменной *UnitNo* (номер модуля): встроенный порт EtherNet/IP или модуль интерфейса EtherNet/IP. Дополнительные сведения см. в разделе *Допустимый диапазон для NewPassword* на стр. 2-1313 ниже.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitNo	Сведения о перечислителях перечислимого типа _eUnitNo см. в разделе *Функция* на стр. 2-1313.																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NewUserName																				ОК
NewPassword																				ОК

Функция

Команда ChangeFTPAccount изменяет имя и пароль для входа на сервер FTP для встроенного порта EtherNet/IP или для модуля интерфейса EtherNet/IP, который указан с помощью параметра *UnitNo* (номер модуля). Новые значения имени и пароля задаются параметрами *NewUserName* (имя для входа) и *NewPassword* (пароль).
Когда вход *Execute* переходит из состояния ЛОЖЬ в состояние ИСТИНА, значения переменных *NewUserName* и *NewPassword* записываются в качестве имени и пароля для входа на сервер FTP для встроенного порта EtherNet/IP.
Выход *Busy* находится в состоянии ИСТИНА во время выполнения команды, а выход *Done* переходит в состояние ИСТИНА, когда завершается прием запроса на изменение настроек.
После перехода выхода *Done* в состояние ИСТИНА настройки, однако, еще не вступают в силу.

Если в параметре *UnitNo* указан модуль интерфейса EtherNet/IP, модуль интерфейса EtherNet/IP перезапускается по окончании выполнения команды. Новые имя и пароль для входа вступают в силу по окончании перезапуска модуля.

Эту команду можно использовать для изменения имени и пароля для входа на сервер FTP для встроенного порта EtherNet/IP или модуля интерфейса EtherNet/IP с терминала HMI.

Для параметра *UnitNo* используется перечислимый тип данных _eUnitNo. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_CBU_CPU	Встроенный порт EtherNet/IP
_CBU_No00..._CBU_No15*1	Номер модуля 00...15 модуля EtherNet/IP

*1. Это значение можно задать только для модуля ЦПУ серии NJ.

Допустимый диапазон для *NewPassword*

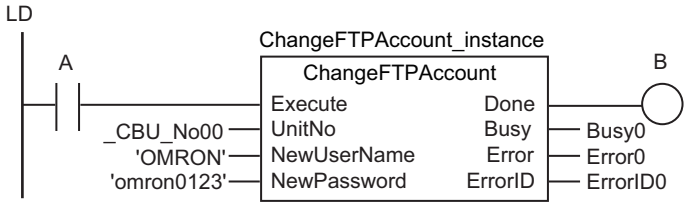
Диапазон допустимых значений переменной *NewPassword* зависит от того, что указано в переменной *UnitNo* (номер модуля): встроенный порт EtherNet/IP или модуль интерфейса EtherNet/IP. См. таблицу ниже.

Настройка *UnitNo*	Диапазон допустимых значений
Встроенный порт EtherNet/IP	От 8 до 32 однобайтовых буквенно-цифровых символов (с различением строчных и заглавных букв)
Модуль интерфейса Ethernet/IP*1	От 1 до 8 однобайтовых буквенно-цифровых символов (с различением строчных и заглавных букв)

*1. Это значение можно задать только для модуля ЦПУ серии NJ.

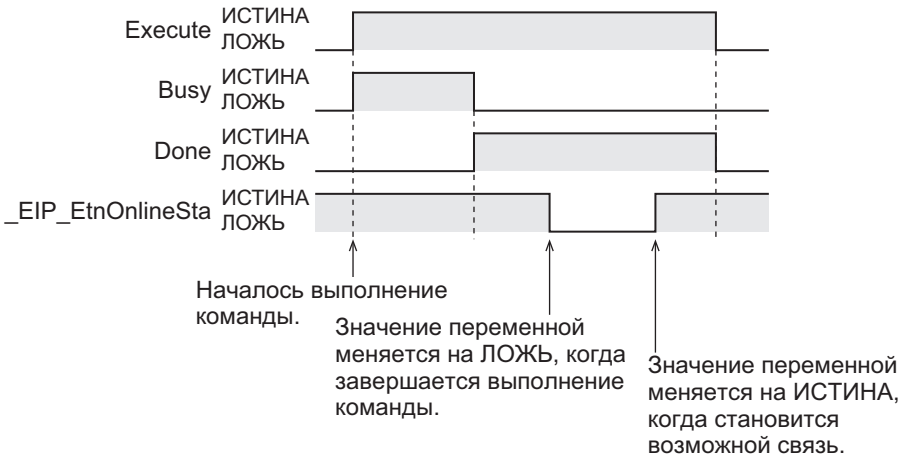
Пример записи

Ниже показан пример использования команды для изменения имени и пароля для входа на сервер FTP для модуля интерфейса EtherNet/IP с номером модуля 00.
Когда значение переменной A (*Execute*) меняется на ИСТИНА с терминала HMI или другого устройства, имя для входа на сервер FTP меняется на «OMRON», а пароль меняется на «omron0123».




```
ChangeFTPAccount_instance(A,_CBU_No00,'OMRON','omron0123',B,Busy0,Error0,ErrorID0);
```

Для модуля EtherNet/IP с номером модуля 00 производятся следующие изменения:
имя для входа на сервер FTP меняется на «OMRON», а пароль меняется на «omron0123».



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

- Изменение имени и пароля для входа на сервер FTP с помощью этой команды возможно, даже если модуль ЦПУ защищен от записи.
- Если имя и пароль для входа на сервер FTP будут изменены с помощью этой команды во время передачи файла, передача файла будет продолжена.
- Если учетные данные для входа на сервер FTP будут изменены с помощью этой команды после выполнения входа на сервер FTP, уже установленные сеансы FTP продолжат работать даже после изменений.

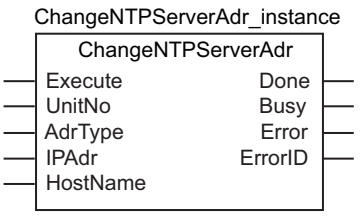
Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду невозможно использовать в событийной задаче. В этом случае произойдет ошибка компиляции.
- При возникновении ошибки состояние выхода *Error* меняется на ИСТИНА. В следующей таблице приведены возможные значения переменной *ErrorID* и поясняется их значение.

Значение *ErrorID*	Наименование ошибки	Описание
16#0400	Входное значение вне диапазона	• Значение входной переменной находится за пределами допустимого диапазона. • Неверное значение входной переменной.
16#2400	Нет права на выполнение	Команда была выполнена, когда изменение состояния было невозможно: • Когда уже выполнялась другая операция изменения параметров. • Когда производился перезапуск встроенного порта EtherNet/IP. • Когда выполнялась загрузка параметров теговых логических связей из Network Configurator.
16#2402	Слишком много одновременно выполняемых команд	Одновременно выполнялось слишком много команд ChangeIPAdr, ChangeFTPAccount и ChangeNTPServerAdr.

ChangeNTPServerAdr

Команда ChangeNTPServerAdr изменяет адрес сервера NTP встроенного порта EtherNet/IP модуля ЦПУ или адрес сервера NTP модуля интерфейса EtherNet/IP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ChangeNTPServerAdr	Изменение адреса сервера NTP	FB		ChangeNTPServerAdr_instance(Execute, UnitNo, AdrType, IPAdr, HostName, Done, Busy, Error, ErrorID);

Информация о версии

Для использования этой команды требуется модуль ЦПУ с версией модуля 1.02 или более поздней и Sysmac Studio версии 1.03 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
UnitNo	Номер модуля	Вход	Номер модуля, для которого нужно изменить адрес сервера NTP	_CBU_CPU или _CBU_No00..._CBU_No15 *1	---	_CBU_No00
AdrType	Способ настройки сервера		Способ настройки адреса сервера NTP ИСТИНА: имя хоста ЛОЖЬ: IP-адрес	Зависит от типа данных.		ЛОЖЬ
IPAdr[] (массив)*2	IP-адрес		IP-адрес сервера NTP	*3		---
HostName	Имя хоста		Имя хоста адреса сервера NTP	От 1 до 200 однобайтовых буквенно-цифровых символов, "-" (дефис) и "." (точка)*4		

*1. Значение *_CBU_No00..._CBU_No15* можно задать только для модуля ЦПУ серии NJ.
*2. Это массив с четырьмя элементами с номерами от 0 до 3.
*3. Допустимый диапазон зависит от того, что указано в переменной *UnitNo* (номер модуля): встроенный порт EtherNet/IP или модуль интерфейса EtherNet/IP. Дополнительные сведения см. в разделе *Допустимый диапазон для IPAdr[]* на стр. 2-1317 ниже.
*4. Между символами "." (точка) и "." (точка) может находиться от 1 до 63 однобайтовых буквенно-цифровых символов. Диапазон допустимых значений *HostName* действителен, только если для переменной *AdrType* (способ настройки сервера) установлено значение ЛОЖЬ.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
UnitNo	Сведения о перечислителях перечислимого типа _eUnitNo см. в разделе *Функция* на стр. 2-1317.																			
AdrType	ОК																			
IPAdr[] (массив)		ОК																		
	Следует указать массив.																			
HostName																				ОК

Функция

Команда ChangeNTPServerAdr изменяет адрес сервера NTP встроенного порта EtherNet/IP или модуля интерфейса EtherNet/IP, который указан с помощью параметра *UnitNo* (номер модуля).
Если для параметра *AdrType* (способ настройки сервера) установлено значение ИСТИНА, адрес сервера NTP изменяется на IP-адрес IPAdr[].
Если же для параметра *AdrType* (способ настройки сервера) установлено значение ЛОЖЬ, адрес сервера NTP изменяется на имя хоста *HostName*.
Когда вход *Execute* переходит из состояния ЛОЖЬ в состояние ИСТИНА, значение переменной IPAdr[] или *HostName* записывается в качестве адреса сервера NTP. Выход *Busy* находится в состоянии ИСТИНА во время выполнения команды, а выход *Done* переходит в состояние ИСТИНА, когда завершается прием запроса на изменение настроек.
После перехода выхода *Done* в состояние ИСТИНА настройки, однако, еще не вступают в силу.

Если в параметре *UnitNo* указан модуль интерфейса EtherNet/IP, модуль интерфейса EtherNet/IP перезапускается по окончании выполнения команды. Новый адрес сервера NTP вступает в силу по окончании перезапуска модуля.

Эту команду можно использовать для изменения адреса сервера NTP встроенного порта EtherNet/IP или модуля интерфейса EtherNet/IP с терминала HMI.

Для параметра *UnitNo* используется перечислимый тип данных _eUnitNo. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_CBU_CPU	Встроенный порт EtherNet/IP
_CBU_No00..._CBU_No15*1	Номер модуля 00...15 модуля EtherNet/IP

*1. Это значение можно задать только для модуля ЦПУ серии NJ.

Значение IP-адреса задается с помощью элементов 0...3 массива IPAdr[]. Например, для изменения адреса сервера NTP на IP-адрес 130.58.17.32 в элементы массива IPAdr[] нужно записать следующие значения: IPAdr[0] = BYTE#16#82, IPAdr[1] = BYTE#16#3A, IPAdr[2] = BYTE#16#11 и IPAdr[3] = BYTE#16#20.

Допустимый диапазон для IPAdr[]

Диапазон допустимых значений переменной IPAdr[] зависит от того, что указано в переменной *UnitNo* (номер модуля): встроенный порт EtherNet/IP или модуль интерфейса EtherNet/IP. См.

таблицу ниже. Диапазоны допустимых значений действительны, только если значение переменной *AdrType* = ИСТИНА.

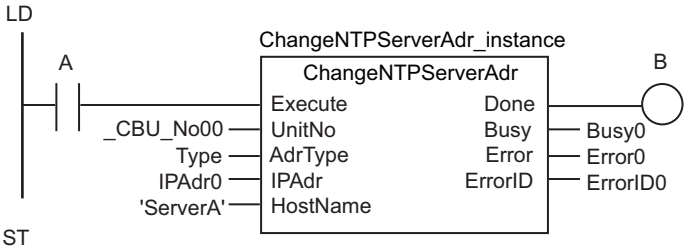
Настройка *UnitNo*	Диапазон допустимых значений
Встроенный порт EtherNet/IP	Перечисленные ниже IP-адреса использовать нельзя. Все остальные IP-адреса действительны. • IP-адреса, начинающиеся со 127, 0 или 255. • IP-адреса класса D (224.0.0.0...239.255.255.255) • IP-адреса класса E (240.0.0.0...255.255.255.255) • IP-адреса, зарезервированные для функции AutoIP*1 (169.254.0.0...169.254.255.255) • IP-адреса портов USB (192.168.255.0...192.168.255.255)
Модуль интерфейса Ethernet/IP	Перечисленные ниже IP-адреса использовать нельзя. Все остальные IP-адреса действительны. • IP-адреса, начинающиеся со 127. • IP-адреса класса D (224.0.0.0...239.255.255.255) • IP-адреса класса E (240.0.0.0...255.255.255.255)

*1. AutoIP — это функция автоматического назначения IP-адреса, предусмотренная в операционной системе Windows 98 и более поздних версиях этой ОС.

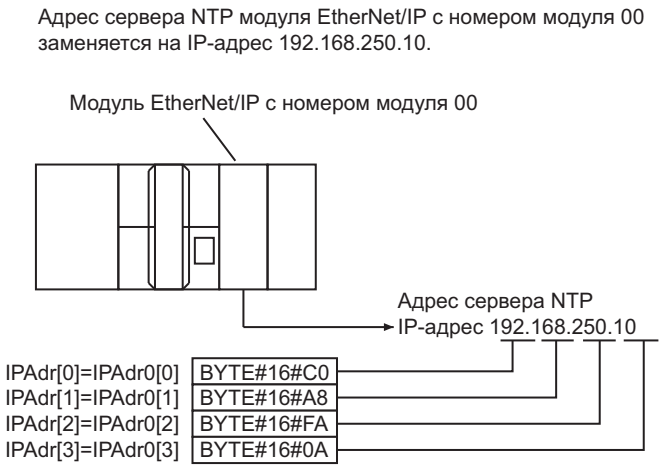
Пример записи

Ниже показан пример использования команды для изменения адреса сервера NTP модуля интерфейса EtherNet/IP с номером модуля 00.

Когда значение переменной A (*Execute*) меняется на ИСТИНА с терминала HMI или другого устройства, адрес сервера NTP изменяется.
Пусть, например, IPAdr0[0] = BYTE#16#C0, IPAdr0[1] = BYTE#16#A8, IPAdr0[2] = BYTE#16#FA, а IPAdr0[3] = BYTE#16#0A.
Если для параметра *AdrType* (способ настройки сервера) установлено значение ИСТИНА, адрес сервера NTP поменяется на IP-адрес 192.168.250.10.
Если же для параметра *AdrType* установлено значение ЛОЖЬ, адрес сервера NTP поменяется на имя хоста ServerA.

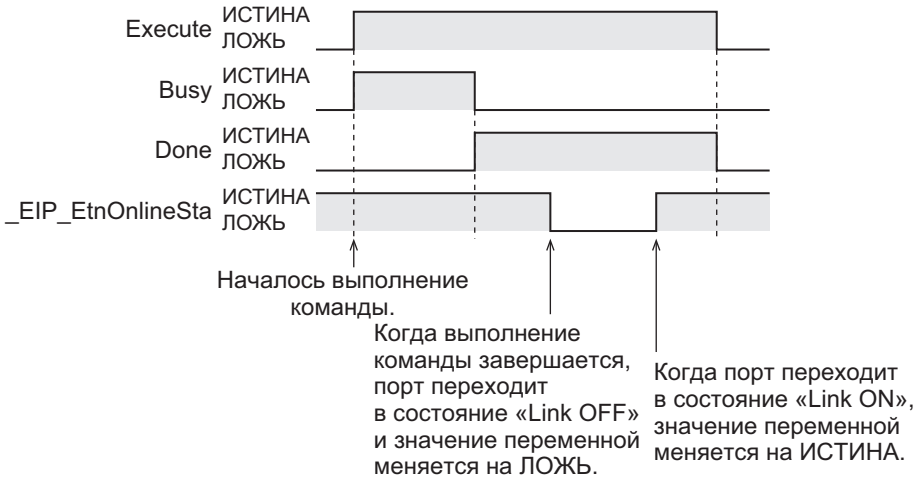



```
ChangeNTPServerAdr_instance(A,_CBU_No00,Type,IPAdr0,'ServerA',B,Busy0,Error0,ErrorID0);
```



С помощью системной переменной *_EIP_EtnOnlineSta*, *_EIP1_EtnOnlineSta* или *_EIP2_EtnOnlineSta* можно проверить, находится ли порт в состоянии «Link ON» («Соединение установлено»).
Здесь в качестве примера используется системная переменная *_EIP_EtnOnlineSta*, но эта информация также относится и к системным переменным *_EIP1_EtnOnlineSta* и *_EIP2_EtnOnlineSta*.

Когда значение *Busy* меняется на ЛОЖЬ, порт переходит в состояние «Link OFF» («Отсутствие соединения»), а значение *_EIP_EtnOnlineSta* меняется на ЛОЖЬ. Когда порт переходит в состояние «Link ON» («Соединение установлено»), значение *_EIP_EtnOnlineSta* меняется на ИСТИНА.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta *1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Дополнительная информация

- Изменение адреса сервера NTP с помощью этой команды возможно, даже если модуль ЦПУ защищен от записи.
- Если сервер NTP, адрес которого изменяется, настроен для работы через заданные интервалы времени, отсчет заданного временного интервала начнется по окончании выполнения этой команды.

Меры предосторожности для обеспечения надлежащей эксплуатации

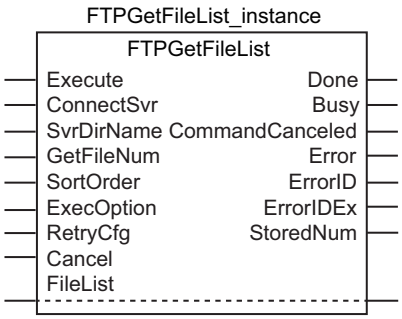
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду невозможно использовать в событийной задаче. В этом случае произойдет ошибка компиляции.
- При возникновении ошибки состояние выхода *Error* меняется на ИСТИНА. В следующей таблице приведены возможные значения переменной *ErrorID* и поясняется их значение.

Значение *ErrorID*	Наименование ошибки	Описание
16#0400	Входное значение вне диапазона	Неверное значение переменной IPAdr[] или *HostName**1
16#2400	Нет права на выполнение	Команда была выполнена, когда изменение состояния было невозможно: • Когда уже выполнялась другая операция изменения параметров. • Когда производился перезапуск встроенного порта EtherNet/IP. • Когда выполнялась загрузка параметров теговых логических связей из Network Configurator.
16#2402	Слишком много одновременно выполняемых команд	Одновременно выполнялось слишком много команд ChangeIPAdr, ChangeFTPAccount и ChangeNTPServerAdr.

*1. Проверка на принадлежность допустимому диапазону выполняется только при соответствующем значении параметра *AdrType*.

FTPGetFileList

Команда FTPGetFileList позволяет получить список файлов, находящихся на сервере FTP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FTPGetFileList	Получение списка файлов на сервере FTP	FB		FTPGetFileList_instance(Execute, ConnectSvr, SvrDirName, GetFileNum, SortOrder, ExecOption, RetryCfg, Cancel, FileList, Done, Busy, CommandCanceled, Error, ErrorID, ErrorIDEx, StoredNum);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
ConnectSvr	Параметры подключенного сервера FTP	Вход	Параметры подключенного сервера FTP	---	---	*1
SvrDirName	Имя каталога сервера FTP		Имя каталога на сервере FTP, для которого нужно получить список файлов	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)*2		"*3
GetFileNum	Количество файлов в списке		Количество файлов в списке	1…1000		1
SortOrder*4	Порядок сортировки		Порядок сортировки списка файлов	_NAME_ASC, _NAME_DESC, _DATE_ASC, _DATE_DESC		_NAME_ASC
ExecOption	Параметры выполнения команды для FTP		Дополнительные параметры выполнения команды для FTP	---		---
RetryCfg	Настройка повтора выполнения		Настройка повторных попыток выполнения команды			
Cancel	Отмена		ИСТИНА: выполнение команды отменено. ЛОЖЬ: выполнение команды не отменено.	Зависит от типа данных.		ЛОЖЬ
FileList[] (массив)*5*6*7	Подробные сведения о файлах	Вход-выход	Полученный список файлов с подробными сведениями о них	---	---	*1
CommandCanceled	Отмена завершена	Выход	ИСТИНА: отмена завершена. ЛОЖЬ: отмена не завершена.	Зависит от типа данных.	---	---
StoredNum	Количество файлов в полученном списке		Количество файлов, по которым получены сведения	0…1000		

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

*2. В именах каталогов сервера FTP нельзя использовать следующие символы:
* ? < > | "

*3. При входе на сервер FTP по умолчанию используется домашний каталог.

*4. Если сервер FTP не поддерживает сортировку по именам, имена упорядочиваются по возрастанию независимо от значения *SortOrder*.

*5. Массив может содержать не более 1000 элементов.

*6. Это одномерный массив. Если будет указан массив с двумя или большим числом измерений, произойдет ошибка сборки.

*7. Номер первого элемента массива: 0. Если для первого элемента массива будет указано число, отличное от 0, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ConnectSvr	Подробные сведения о структуре _sFTP_CONNECT_SVR см. в разделе *Функция* на стр. 2-1323.																			
SvrDirName																				ОК
GetFileNum							ОК													
SortOrder	Сведения о перечислителях перечислимого типа _eFILE_SORT_ORDER см. в разделе *Функция* на стр. 2-1323.																			
ExecOption	Подробные сведения о структуре _sFTP_EXEC_OPTION см. в разделе *Настройка дополнительных параметров команды для сервера FTP* на стр. 2-1325.																			
RetryCfg	Подробные сведения о структуре _sFTP_RETRY_CFG см. в разделе *Настройка выполнения повторных попыток соединения с сервером FTP* на стр. 2-1325.																			
Cancel	ОК																			
FileList[] (массив)	Подробные сведения о структуре _sFTP_FILE_DETAIL см. в разделе *Функция* на стр. 2-1323.																			
CommandCanceled	ОК																			
StoredNum							ОК													

Функция

Команда FTPGetFileList позволяет получить список файлов и сведения о файлах, расположенных в каталоге с указанным именем (*SvrDirName*) на подключенном сервере FTP (*ConnectSvr*). Количество файлов, список и сведения о которых нужно получить, указывается в переменной *GetFileNum*. С помощью переменной *SortOrder* можно задать порядок сортировки полученной информации о файлах.

Для переменной *ConnectSvr* используется структурный тип данных _sFTP_CONNECT_SVR. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
ConnectSvr	Параметры подключенного сервера FTP	Параметры подключенного сервера FTP	_sFTP_CONNECT_SVR	---	---	---
Adr	Адрес	IP-адрес или имя хоста*1	STRING	1...200 байт*2	---	---
PortNo	Номер порта	Номер порта TCP сервера FTP для соединения	UINT	0...65535*3		
UserName	Имя пользователя	Имя пользователя на сервере FTP	STRING	Макс. 33 байт*4*5*6		
Password	Пароль	Пароль для входа на сервер FTP	STRING	Макс. 33 байт*4*5*6		

*1. Чтобы можно было указать имя хоста, должны быть отдельно настроены параметры DNS или хостов.
*2. Можно использовать следующие однобайтовые символы: "A...Z", "a...z", "0...9", "-" (дефис), "." (точка) и "_" (подчеркивание).
*3. Если указано значение 0, используется порт TCP с номером 21.
*4. Можно использовать следующие однобайтовые символы: "A...Z", "a...z", "0...9", "-" (дефис), "." (точка) и "_" (подчеркивание). Для модуля ЦПУ с версией модуля 1.16 или более поздней также можно использовать символы "\" (обратная косая черта) и "@".
*5. При подсчете количества байтов также следует учитывать нулевой символ (NULL).
*6. В случае модулей ЦПУ с версией модуля 1.08 следует указать текстовую строку, состоящую из одного или более символов. Если будет указана текстовая строка, содержащая только завершающий пустой символ (NULL), произойдет ошибка.

Для параметра *SortOrder* используется перечислимый тип данных _eFILE_SORT_ORDER. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_NAME_ASC	По возрастанию в алфавитном порядке (по именам)
_NAME_DESC	По убыванию в алфавитном порядке (по именам)
_DATE_ASC	По возрастанию даты последнего изменения
_DATE_DESC	По убыванию даты последнего изменения

Подробные сведения о файлах сохраняются в переменную FileList[]. Количество файлов, по которым была получена информация, записывается в переменную *StoredNum*.

Для переменной FileList[] используется структурный тип данных _sFTP_FILE_DETAIL. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
FileList	Подробные сведения о файлах	Полученный список файлов с подробными сведениями о них	_sFTP_FILE_DETAIL	---	---	---
Name	Имя файла или папки	Имя файла или папки	STRING	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	---
ModifiedDate	Дата последнего изменения	Дата последнего изменения файла или папки	DATE_AND_TIME	---	---	
Size	Размер файла	Размер файла*1	ULINT		Байты	
ReadOnly	Атрибут «только для чтения»	Атрибут «только для чтения» файла или папки ИСТИНА: только чтение ЛОЖЬ: не только чтение	BOOL	Зависит от типа данных.	---	
Folder	Папка	ИСТИНА: папка ЛОЖЬ: не папка	BOOL			

*1. Для папки размер файла равен 0.

Настройка дополнительных параметров команды для сервера FTP

С помощью переменной *ExecOption* для команды получения списка файлов с сервера FTP можно задать ряд дополнительных параметров, определяющих особенности ее выполнения. Предусмотрены те же параметры, что и для команды FTPGetFile. Подробную информацию см. в описании команды *FTPGetFile* на стр. 2-1337.
Однако для этой команды действителен только параметр *ExecOption.PassiveMode*.

Настройка выполнения повторных попыток соединения с сервером FTP

Если соединение с сервером FTP не удается успешно установить в течение указанного времени ожидания *RetryCfg.TimeOut*, операция подключения к серверу FTP завершается из-за превышения времени ожидания. Если сервер FTP отклоняет соединение, операция завершается до истечения времени ожидания.
После неудачной попытки установить соединение операция подключения выполняется повторно по истечении заданного интервала повторных попыток *RetryInterval*.
Если соединение с сервером FTP не удается установить за количество повторных попыток, заданное параметром *RetryCfg.RetryNum*, происходит ошибка выполнения команды.

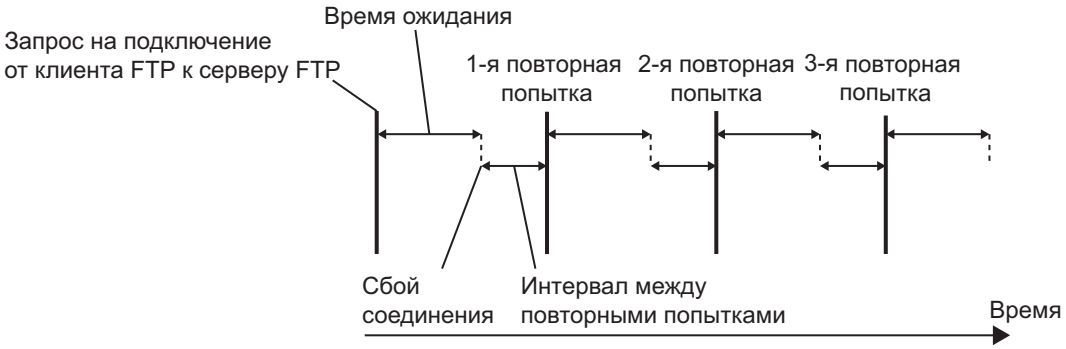
Если после успешного подключения к серверу FTP в сети возникает проблема, из-за которой передача файлов прерывается на время, превышающее время, указанное в *RetryCfg.TimeOut*, происходит тайм-аут и повторных попыток не предпринимается.

Для переменной *RetryCfg* используется структурный тип данных _sFTP_RETRY_CFG. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
RetryCfg	Настройка повтора выполнения	Настройка повторных попыток выполнения команды	_sFTP_RETRY_CFG	---	---	---
TimeOut	Время ожидания	Время ожидания установления соединения с сервером FTP	UINT	0...60*1	Секунды	20
RetryNum	Количество повторных попыток	Количество повторных попыток установить соединение в случае сбоя при установлении соединения	UINT	0…3	Кол-во раз	0
RetryInterval	Интервал повторных попыток	Интервал между повторными попытками установить соединение в случае сбоя при установлении соединения	UINT	0...65535*2	Секунды	1

*1. Если задано значение *0*, время ожидания принимается равным 20 с.
*2. Если задано значение *0*, интервал повторных попыток принимается равным 1 с.

На следующем рисунке показана взаимосвязь между временем ожидания, количеством повторных попыток и интервалом повторных попыток для случая, когда клиент FTP выполняет операцию подключения к серверу FTP.

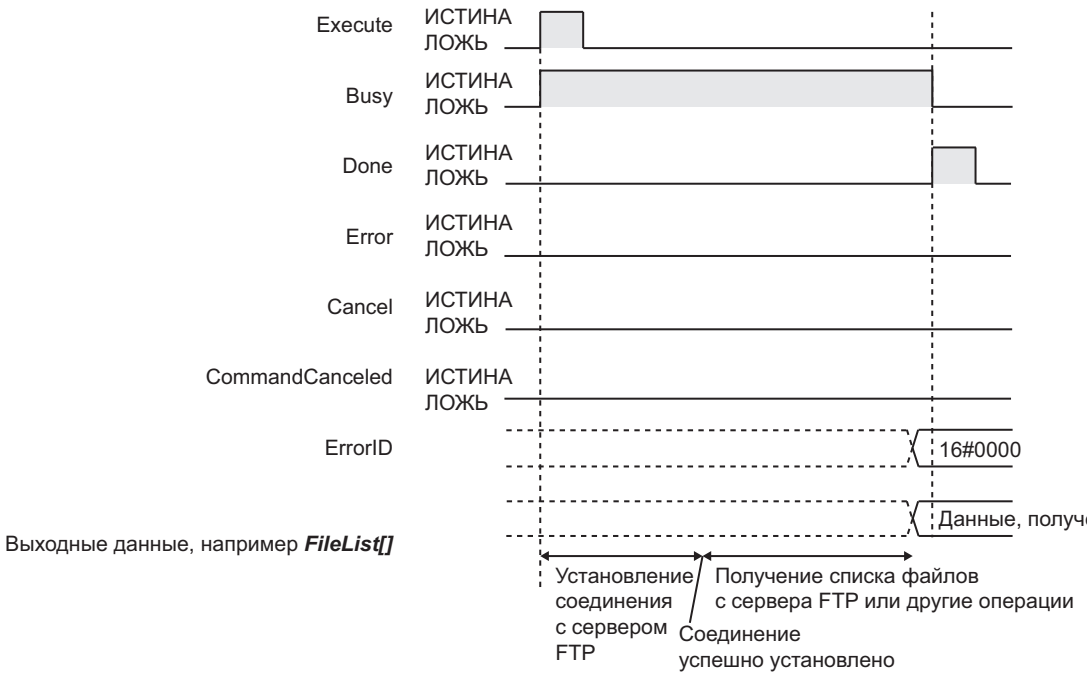


● Успешное подключение к серверу FTP

После того как соединение с сервером FTP успешно установлено и от сервера FTP получен список файлов, происходит следующее:

- В переменную *ErrorID* записывается значение 16#0000.
- Полученные данные сохраняются в качестве выходных данных в переменную FileList[].
- Выход *Done* переходит в состояние ИСТИНА.

На рисунке ниже показан пример временной диаграммы для случая, когда операция подключения к серверу FTP завершается успешно.

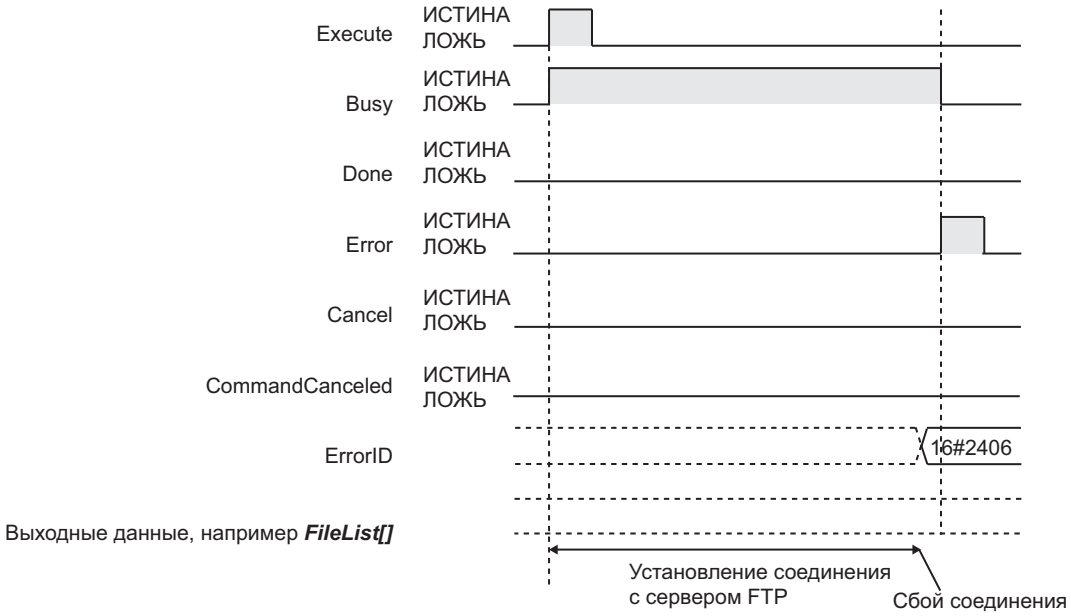


● Сбой подключения к серверу FTP

Если попытка подключиться к серверу FTP завершается неудачей, происходит следующее:

- В переменную *ErrorID* сохраняется код ошибки.
- Выход *Error* переходит в состояние ИСТИНА.

На рисунке ниже показан пример временной диаграммы для случая, когда операция подключения к серверу FTP завершается сбоем.



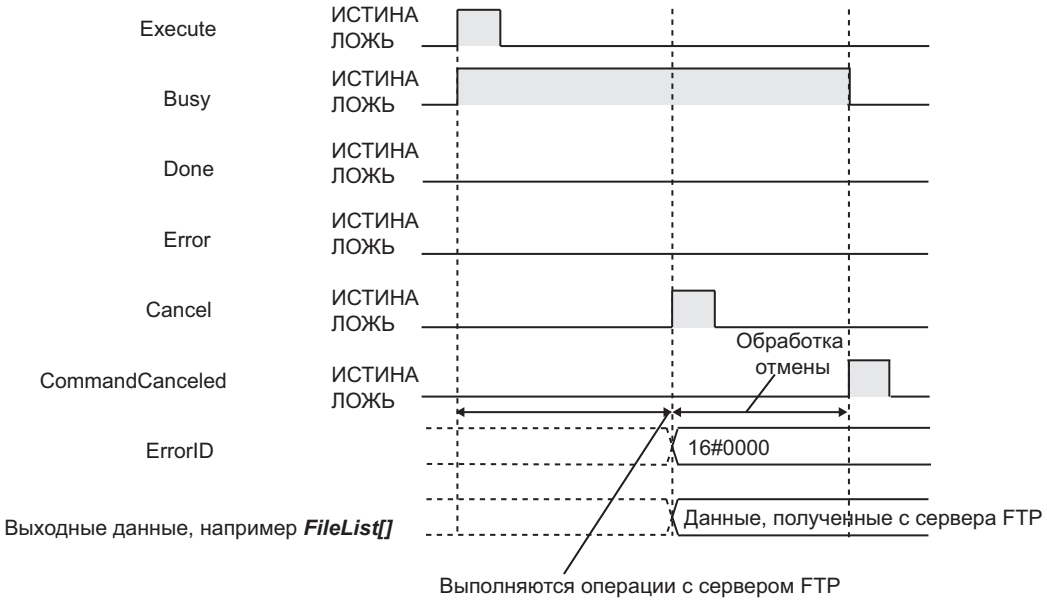
Отмена выполнения команды

Если во время выполнения команды значение переменной *Cancel* меняется на ИСТИНА, операции с сервером FTP принудительно завершаются.

Это можно использовать для завершения работы команды, когда получение списка файлов или установление соединения с сервером FTP длится слишком долго.

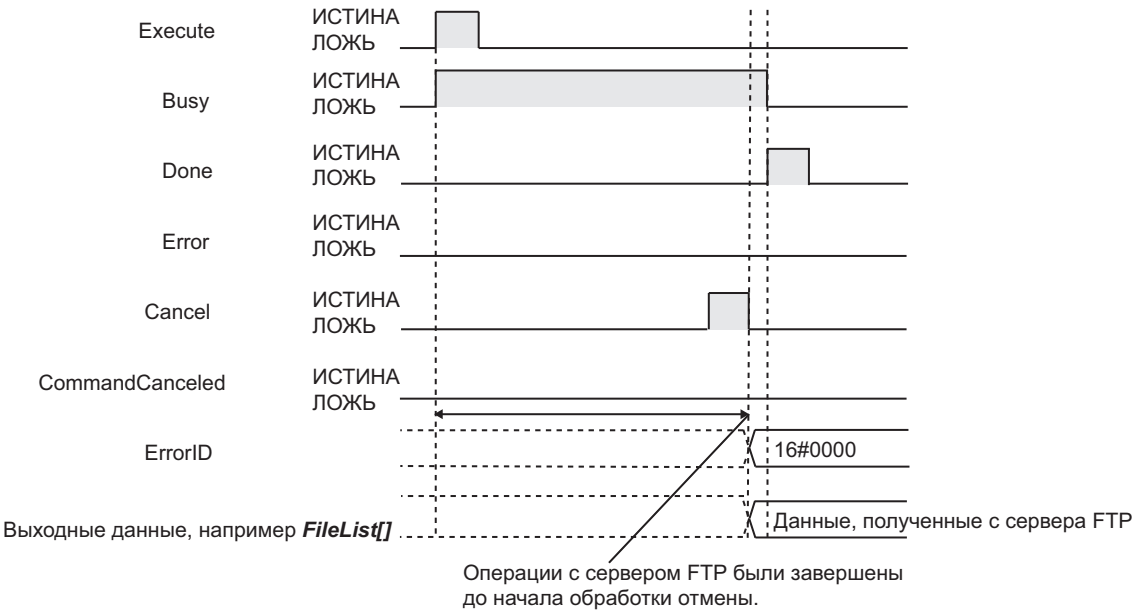
● Значение *Cancel* меняется на ИСТИНА во время выполнения операций с сервером FTP

Если вход *Cancel* переходит в состояние ИСТИНА, когда команда FTPGetFileList получает список файлов от сервера FTP, происходит следующее:
Все сведения о файлах, полученные от сервера FTP, сохраняются в переменную FileList[].
В переменную *StoredNum* записывается количество файлов, для которых были корректно получены сведения.
Выход *Done* не переходит в состояние ИСТИНА.
По завершении отмены значение *CommandCanceled* меняется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения отмены.



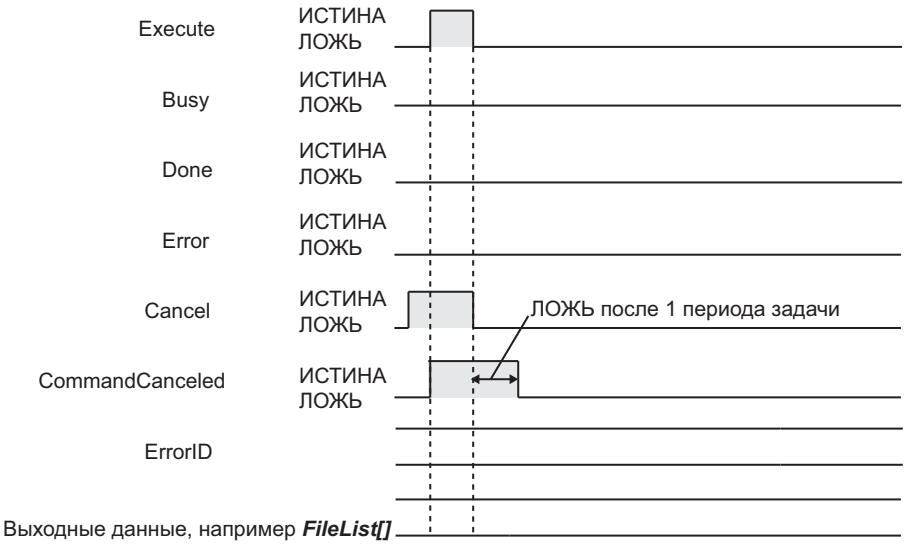
● Операции с сервером FTP были завершены до начала обработки отмены

Если операции с сервером FTP уже завершены к моменту перехода входа *Cancel* в состояние ИСТИНА (т. е. еще до начала выполнения отмены), выход *Done* в этом случае переходит в состояние ИСТИНА, указывая на нормальное завершение команды. Значение *CommandCanceled* не меняется на ИСТИНА.



● Состояние ИСТИНА одновременно на входах *Cancel* и *Execute*

Если оба входа, *Cancel* и *Execute*, одновременно находятся в состоянии ИСТИНА, приоритетом обладает операция отмены, поэтому операции с сервером FTP не выполняются. Переменная *CommandCanceled* переходит в состояние ИСТИНА.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.

*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если в каталоге, который указан входной переменной *SvrDirName*, отсутствуют файлы или подкаталоги, выход *Done* переходит в состояние ИСТИНА, указывая на нормальное завершение команды. Если в переменную *StoredNum* записывается значение 0, в переменную FileList[] ничего не сохраняется.
- Если количество элементов в массиве FileList[] меньше количества файлов, указанного с помощью входной переменной *GetFileNum*, сохраняется только информация о файлах, которая может поместиться в массив FileList[], а не помещающаяся информация о файлах не сохраняется. Выход *Error* в этом случае в состояние ИСТИНА не переходит.
- Если длина имени файла превышает 255 символов, в элемент *Name* структуры FileList[] сохраняются только первые 255 символов. Выход *Error* в этом случае в состояние ИСТИНА не переходит.
- В зависимости от особенностей сервера FTP может оказаться невозможным получить сведения о некоторых или обо всех файлах. Если не удалось получить сведения о каком-либо файле, члены структуры в соответствующем элементе массива FileList[] принимают указанные ниже значения. Значение переменной *Error* в этом случае меняется на ЛОЖЬ.

Член структуры	Значение
ModifiedDate	DT#1970-01-01-00:00:00.000000000
Size	0
ReadOnly	ЛОЖЬ
Folder	ЛОЖЬ

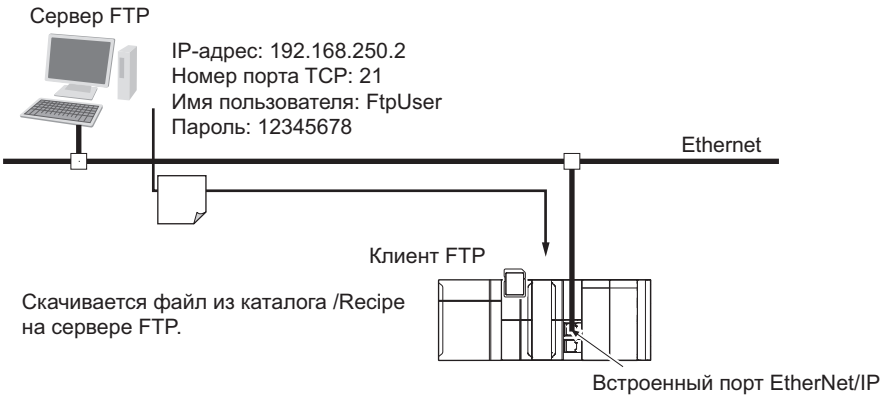
- Допускается одновременное выполнение не более 3 экземпляров следующих команд: FTPGetFileList, FTPGetFile, FTPPutFile, FTPRemoveFile и FTPRemoveDir.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение какого-либо входного параметра находится за пределами допустимого диапазона.
  b) ".." указано для уровня каталога в *SvrDirName*.
  c) Для *SvrDirName* указан неверный путь, например "//".
  d) На сервере FTP отсутствует каталог с указанным именем *SvrDirName*.
  e) Сервер FTP, указанный параметром *ConnectSvr*, отсутствует в сети или не работает.
  f) Было выполнено более 3 следующих команд одновременно: FTPGetFileList, FTPGetFile, FTPPutFile, FTPRemoveFile и FTPRemoveDir.
  g) Операция передачи файлов была прервана во время операции подключения к серверу FTP из-за проблемы в сети.
- Для этой команды дополнительный код ошибки *ErrorIDEx* содержит код ответа FTP, который был возвращен сервером FTP. В следующей таблице перечислены типичные значения дополнительного кода ошибки *ErrorIDEx*, описано значение ошибок и приведены возможные способы их устранения. Дополнительные сведения см. в документации по серверу FTP.

Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = WORD#16#2407.

Значение *ErrorIDEx*	Значение	Способ устранения
16#000001A9	Не удалось установить соединение для передачи данных.	В случае использования протокола FTP для обмена данными с сервером FTP через Интернет проследите, чтобы для параметра Open mode (Режим открытия) для FTP не было установлено значение Active (Активный).
16#000001AA	Соединение было закрыто. Передача данных была прервана.	Проверьте соединение с сервером FTP. Убедитесь, что сервер FTP работает.
16#000001C2	Не удалось выполнить запрошенную операцию с файлом. Использовать файл было невозможно, например, из-за того, что он уже был открыт.	Убедитесь, что файл не открыт для какого-нибудь другого приложения.
16#00000212	Вход пользователя на сервер был невозможен.	Проверьте имя пользователя и пароль для входа на сервер FTP.
16#00000214	Для сохранения файлов требуется учетная запись.	Проверьте наличие у пользователя прав доступа к FTP.
16#00000226	Выполнить запрошенную операцию с файлом было невозможно, так как было невозможно использовать файл (например, не удалось получить доступ к файлу из-за того, что файл не был найден).	Убедитесь, что файл с указанным именем имеется в соответствующем каталоге на сервере FTP. Проверьте права доступа к указанному файлу.
16#00000229	Выполнение было невозможно, так как имя файла было неверным.	Проверьте права доступа к указанному каталогу.

Пример программы

В качестве примера рассмотрим программу, которая загружает файл из каталога «/Recipe» на сервере FTP и сохраняет его в корневой каталог карты памяти SD.
Загружается файл, который располагается в конце списка файлов в каталоге «/Recipe» на сервере FTP, когда файлы упорядочиваются по именам в порядке возрастания.



Контроллер подключен к серверу FTP по сети EtherNet/IP. Значения параметров для подключения к серверу FTP приведены в следующей таблице.

Параметр	Значение
IP-адрес	192.168.250.2
Номер порта TCP	21
Имя пользователя	FtpUser

Параметр	Значение
Пароль	12345678

Соблюдается приведенный ниже порядок действий.

1 Используется команда FTPGetFileList для получения списка файлов с сервера FTP.
В приведенной ниже таблице указаны: имя каталога на сервере FTP, количество файлов в списке, порядок сортировки и переменная для хранения сведений о файлах.

Параметр	Значение
Имя каталога сервера FTP	«/Recipe»
Количество файлов в списке	1000
Порядок сортировки	По возрастанию в алфавитном порядке (по именам)
Переменная для хранения сведений о файлах	FTPFileList[]

2 Используется команда FTPGetFile для загрузки файла, находящегося в конце списка файлов, который был получен на шаге 1 (файлы в списке отсортированы по возрастанию в алфавитном порядке).
Файл сохраняется в корневой каталог на карте памяти SD.

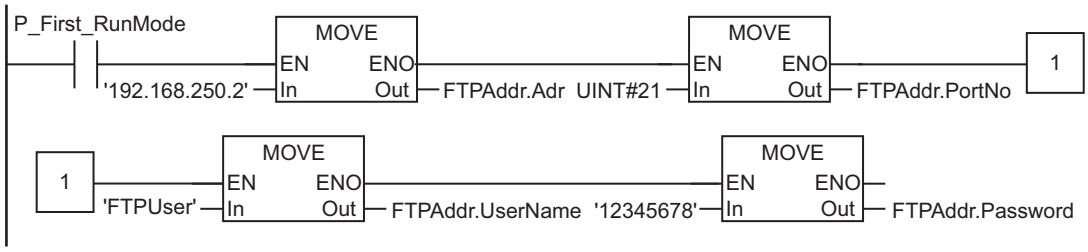
3 Если все операции завершаются нормально, выполняется обработка нормального завершения.
А если возникает ошибка, то выполняется обработка для завершения с ошибкой.

Программа на языке LD

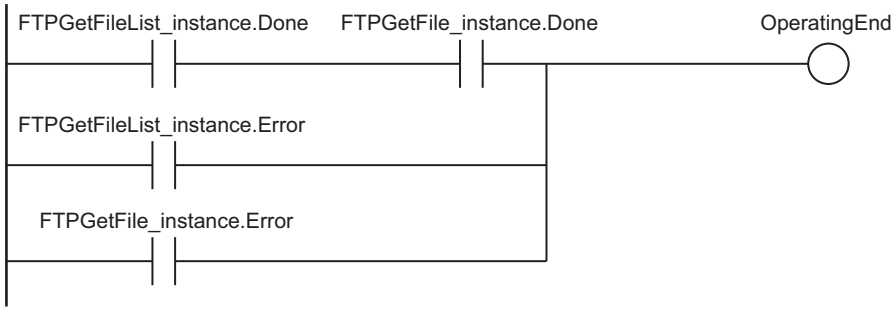
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	FTPGetFileList_instance	FTPGetFileList		Экземпляр команды FTPGetFileList
	FTPGetFile_instance	FTPGetFile		Экземпляр команды FTPGetFile
	FTPAddr	_sFTP_CONNECT_SVR	(Adr := '', PortNo := 0, UserName := '', Password := '')	Параметры подключенного сервера FTP
	FTPFileList	ARRAY[0..999] OF _sFTP_FILE_DETAIL	[1000((Name := '', ModifiedDate := DT#1970-01-01-00:00:00, Size := 0, ReadOnly := False, Folder := False))]	Подробные сведения о файлах
	GetResult	ARRAY[0..0] OF _sFTP_FILE_RESULT	[(Name := '', TxError := False, RemoveError := False, Reserved := [4(16#0)])]	Результаты загрузки файла
	FTPStoredNum	UINT	0	Количество файлов в полученном списке файлов
	LastFileIndex	UINT	0	Индекс последнего файла, когда список упорядочен по именам в порядке возрастания
	RS_instance	RS		Экземпляр команды RS
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка

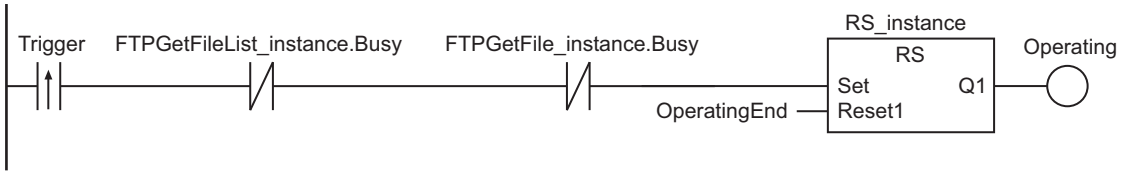
Подготовка параметров подключенного сервера FTP.

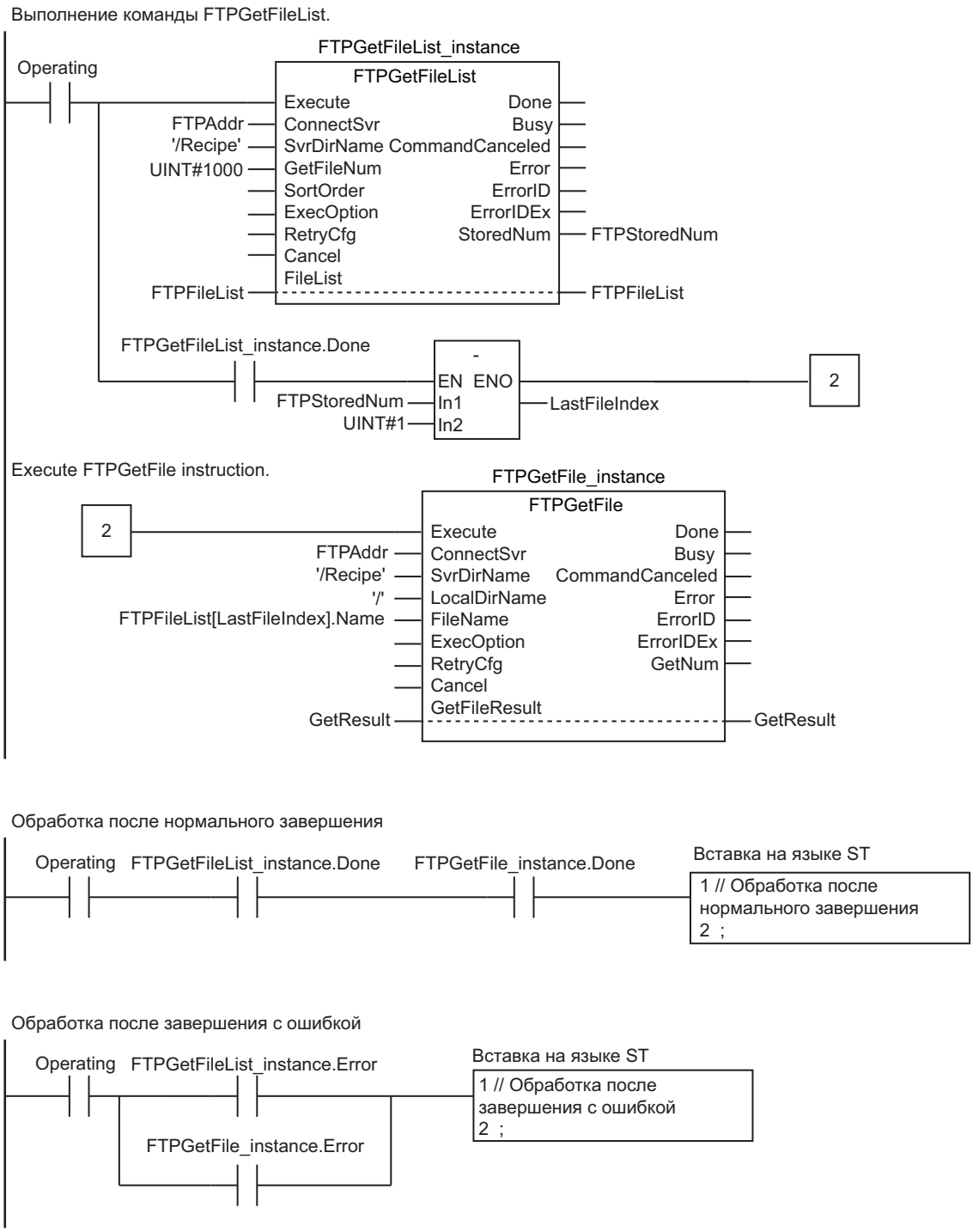


Определение, завершено ли выполнение команды.



Прием условия выполнения.





Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	R_TRIG_instance	R_TRIG		Экземпляр команды R_TRIG
	UP_Q	BOOL	ЛОЖЬ	Выход условия запуска
	FTPGetFile_instance	FTPGetFile		Экземпляр команды FTPGetFile

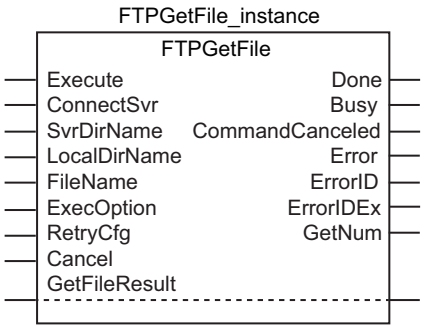
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	FTPGetFileList_instance	FTPGetFileList		Экземпляр команды FTPGetFileList
	FTPFileList	ARRAY[0..999] OF _sFTP_FILE_DETAIL	[1000((Name := '', ModifiedDate := DT#1970-01-01-00:00:00, Size := 0, ReadOnly := False, Folder := False))]	Подробные сведения о файлах
	FTPStoredNum	UINT	0	Количество файлов в полученном списке файлов
	DoFTPTrigger	BOOL	ЛОЖЬ	Условие выполнения для команд FTPGetFileList и FTPGetFile
	FTPAddr	_sFTP_CONNECT_SVR	(Adr := '', PortNo := 0, UserName := '', Password := '')	Параметры подключенного сервера FTP
	GetResult	ARRAY[0..0] OF _sFTP_FILE_RESULT	[(Name := '', TxError := False, RemoveError := False, Reserved := [4(16#0)])]	Результаты загрузки файла
	Stage	UINT	0	Этап выполнения команды
	Trigger	BOOL	ЛОЖЬ	Условие выполнения

```
// Подготовка параметров подключенного сервера FTP.
IF P_First_RunMode THEN
   FTPAddr.Adr := '192.168.250.2';  // Адрес
   FTPAddr.PortNo := UINT#21;  // Номер порта
   FTPAddr.UserName := 'FtpUser';  // Имя пользователя
   FTPAddr.Password := '12345678';  // Пароль
END_IF;

// Прием условия выполнения.
R_TRIG_instance(Trigger, UP_Q);
IF ( (UP_Q = TRUE) AND (FTPGetFileList_instance.Busy = FALSE) AND
     (FTPGetFile_instance.Busy = FALSE) ) THEN
   DoFTPTrigger := TRUE;
   Stage := INT#1;
   FTPGetFileList_instance(  // Инициализация экземпляра.
      Execute := FALSE,
      ConnectSvr := FTPAddr,
      SvrDirName := '/Recipe',
      GetFileNum := UINT#1000,
      FileList := FTPFileList,
      StoredNum => FTPStoredNum) ;
   FTPGetFile_instance(  // Инициализация экземпляра.
      Execute := FALSE,
      ConnectSvr := FTPAddr,
      SvrDirName := '/Recipe',
      LocalDirName := '/',
```

```
      FileName := '',
      GetFileResult := GetResult) ;
END_IF;
IF (DoFTPTrigger = TRUE) THEN
   CASE Stage OF
      1 :  // Выполнение команды FTPGetFileList
         FTPGetFileList_instance(
            Execute := TRUE,  // Выполнение
            ConnectSvr := FTPAddr,  // Подключенный сервер FTP
            SvrDirName := '/Recipe',  // Имя каталога на сервере FTP
            GetFileNum := UINT#1000,  // Количество файлов в списке
            FileList := FTPFileList,  // Сведения о файлах
            StoredNum => FTPStoredNum);  // Количество файлов в полученном списке
         IF (FTPGetFileList_instance.Done = TRUE) THEN
            Stage := INT#2;  // К следующему этапу
         ELSIF (FTPGetFileList_instance.Error = TRUE) THEN
            Stage := INT#10;  // Завершение с ошибкой
         END_IF;
      2 : // Выполнение команды FTPGetFile.
         FTPGetFile_instance(
            Execute := TRUE,  // Выполнение
            ConnectSvr := FTPAddr,  // Подключенный сервер FTP
            SvrDirName := '/Recipe',  // Имя каталога на сервере FTP
            LocalDirName := '/',  // Имя локального каталога
            FileName := FTPFileList[FTPStoredNum - 1].Name,  // Имя файла
            GetFileResult := GetResult);  // Результаты загрузки файлов
         IF (FTPGetFile_instance.Done = TRUE) THEN
            Stage := INT#0;  // Нормальное завершение
         ELSIF (FTPGetFile_instance.Error = TRUE) THEN
            Stage := INT#20;  // Завершение с ошибкой
         END_IF;
      0 :  // Обработка после нормального завершения
         DoFTPTrigger := FALSE;
         Trigger := FALSE;
      ELSE  // Обработка после завершения с ошибкой
         DoFTPTrigger := FALSE;
         Trigger := FALSE;
   END_CASE;
END_IF;
```

FTPGetFile

Команда FTPGetFile загружает файл с сервера FTP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FTPGetFile	Получить файл с сервера FTP	FB		FTPGetFile_instance(Execute, ConnectSvr, SvrDirName, LocalDirName, FileName, ExecOption, RetryCfg, Cancel, GetFileResult, Done, Busy, CommandCanceled, Error, ErrorID, ErrorIDEx,GetNum);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
ConnectSvr	Параметры подключенного сервера FTP	Вход	Параметры подключенного сервера FTP	---	---	*1
SvrDirName	Имя каталога сервера FTP		Имя каталога на сервере FTP, из которого нужно загрузить файл	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)*2		"*3
LocalDirName	Имя локального каталога		Имя каталога, в который нужно сохранить файл, загруженный с сервера FTP	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)		«/»
FileName	Имя файла		Имя загружаемого файла*4	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)*5		*1
ExecOption	Параметры выполнения команды для FTP		Дополнительные параметры выполнения команды для FTP	---		---
RetryCfg	Настройка повтора выполнения		Настройка повторных попыток выполнения команды			
Cancel	Отмена		ИСТИНА: выполнение команды отменено. ЛОЖЬ: выполнение команды не отменено.	Зависит от типа данных.		ЛОЖЬ
GetFile Result[] (массив)*6*7*8	Результаты загрузки файла	Вход-выход	Результаты загрузки файла	---	---	*1
CommandCanceled	Отмена завершена	Выход	ИСТИНА: отмена завершена. ЛОЖЬ: отмена не завершена.	Зависит от типа данных.	---	---
GetNum	Количество загружаемых файлов		Количество файлов, которые нужно загрузить	---		

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.
*2. В именах каталогов сервера FTP нельзя использовать следующие символы:
* ? < > | "
*3. При входе на сервер FTP по умолчанию используется домашний каталог.
*4. В именах файлов допускается использовать подстановочные знаки.

*5. В именах файлов нельзя использовать следующий символ:
|
*6. Массив может содержать не более 1000 элементов.
*7. Это одномерный массив. Если будет указан массив с двумя или большим числом измерений, произойдет ошибка сборки.
*8. Номер первого элемента массива: 0. Если для первого элемента массива будет указано число, отличное от 0, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ConnectSvr	Подробные сведения о структуре _sFTP_CONNECT_SVR см. в разделе *Функция* на стр. 2-1339.																			
SvrDirName																				ОК
LocalDirName																				ОК
FileName																				ОК
ExecOption	Подробные сведения о структуре _sFTP_EXEC_OPTION см. в разделе *Настройка дополнительных параметров команды для сервера FTP* на стр. 2-1341.																			
RetryCfg	Подробные сведения о структуре _sFTP_RETRY_CFG см. в разделе *Настройка выполнения повторных попыток соединения с сервером FTP* на стр. 2-1343.																			
Cancel	ОК																			
GetFileResult[] (массив)	Подробные сведения о структуре _sFTP_FILE_RESULT см. в разделе *Функция* на стр. 2-1339.																			
CommandCanceled	ОК																			
GetNum							ОК													

Функция

Команда FTPGetFile загружает файл с указанным именем (*FileName*) из указанного каталога (*SvrDirName*) на подключенном сервере FTP (*ConnectSvr*) в указанный каталог (*LocalDirName*) на карте памяти SD.
Если каталога с указанным именем *LocalDirName* на карте памяти SD не существует, создается новый каталог и указанный файл загружается в него.
В переменной *FileName* допускается использовать подстановочные знаки. Это позволяет загрузить более одного файла за раз.

Результаты загрузки для каждого файла сохраняются в переменную GetFileResult[].
Количество файлов, которые нужно загрузить, указывается в переменной *GetNum*.
В случае использования подстановочного знака в параметре *FileName* следует указать количество всех файлов, имена которых удовлетворяют подстановочному знаку.

Если фактическое количество переданных файлов будет другим, значение переменной *GetFileResult[].TxError* поменяется на ИСТИНА.
Если при удалении исходного файла после его загрузки возникнет ошибка, значение переменной *GetFileResult[].RemoveError* поменяется на ИСТИНА.

Для переменной *ConnectSvr* используется структурный тип данных _sFTP_CONNECT_SVR.
Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
ConnectSvr	Параметры подключенного сервера FTP	Параметры подключенного сервера FTP	_sFTP_CONNECT_SVR	---	---	---
Adr	Адрес	IP-адрес или имя хоста*1	STRING	1...200 байт*2	---	---
PortNo	Номер порта	Номер порта TCP сервера FTP для соединения	UINT	0...65535*3		
UserName	Имя пользователя	Имя пользователя на сервере FTP	STRING	Макс. 33 байт*4*5*6		
Password	Пароль	Пароль для входа на сервер FTP	STRING	Макс. 33 байт*4*5*6		

*1. Чтобы можно было указать имя хоста, должны быть отдельно настроены параметры DNS или хостов.
*2. Можно использовать следующие однобайтовые символы: "A...Z", "a...z", "0...9", "-" (дефис), "." (точка) и "_" (подчеркивание).
*3. Если указано значение *0*, используется порт TCP с номером 21.
*4. Можно использовать следующие однобайтовые символы: "A...Z", "a...z", "0...9", "-" (дефис), "." (точка) и "_" (подчеркивание). Для модуля ЦПУ с версией модуля 1.16 или более поздней также можно использовать символы "\" (обратная косая черта) и "@".
*5. При подсчете количества байтов также следует учитывать нулевой символ (NULL).
*6. В случае модулей ЦПУ с версией модуля 1.08 следует указать текстовую строку, состоящую из одного или более символов. Если будет указана текстовая строка, содержащая только завершающий пустой символ (NULL), произойдет ошибка.

Для переменной GetFileResult[] используется структурный тип данных _sFTP_FILE_RESULT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
GetFileResult	Результаты загрузки файла	Результаты передачи файла	_sFTP_FILE_RESULT	---	---	---
Name	Имя файла*1	Имя переданного файла	STRING	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	---
TxError	Ошибка передачи	ИСТИНА: передача завершилась с ошибкой. ЛОЖЬ: передача завершилась нормально.	BOOL	Зависит от типа данных.		
RemoveError	Ошибка удаления	ИСТИНА: удаление завершилось с ошибкой. ЛОЖЬ: удаление завершилось нормально.	BOOL			
Reserved	Резерв	Зарезервировано для системы.	ARRAY[0..3] OF BYTE	---		0

*1. Включает расширение имени файла.

Использование подстановочных знаков при указании имен файлов

При указании имен загружаемых файлов в переменной *FileName* допускается использовать подстановочные знаки.
В качестве подстановочных знаков можно использовать символы "*" и "?".
"*" представляет один или несколько символов. "?" представляет только один символ.
Ниже приведены примеры использования подстановочных знаков для указания имен файлов.

Предположим, что каталог на сервере FTP имеет следующую файловую структуру.

```
┣DataFiles (указанный каталог)
┃  ┃ LogA01.log
┃  ┃ LogA02.txt
┃  ┃ LogB.log
┃  ┃ LogC.txt
┃  ┃ ControlDataA1.csv
┃  ┃ ControlDataA10.csv
┃  ┃ ControlDataA100.csv
┃  ┃ ControlDataB10.csv
┃  ┃ ControlDataC100.csv
┃  ┃
┃  ┗ControlSubDataFiles (подкаталог)
┃       SubData_A001.txt
┃       SubData_A002.txt
┃
```

В таблице ниже приведены разные способы использования подстановочных знаков и приведены имена файлов, которые при этом оказываются указаны.

Использование подстановочных знаков	Указанные файлы
Log*.log	LogA01.log, LogB.log
Log?.log	LogB.log
Log?.*	LogB.log, LogC.txt
Data	ControlDataA1.csv, ControlDataA10.csv, ControlDataA100.csv, ControlDataB10.csv, ControlDataC100.csv, (ControlSubDataFiles)*1
*	Все файлы, кроме файлов в подкаталоге
.	Все файлы, кроме файлов в подкаталоге
?.?	Файлы не указаны
????.???	LogB.log, LogC.txt

*1. В зависимости от особенностей сервера FTP также могут включаться файлы в подкаталогах.

При использовании подстановочных знаков можно загрузить не более 1000 файлов.
Если в результате загрузки файлов флаг ошибки *GetFileResult[].TxError* или *GetFileResult[].RemoveError* перейдет в состояние ИСТИНА, произойдет следующее: выход *Error* также перейдет в состояние ИСТИНА, в переменную *ErrorID* будет сохранен соответствующий код первой возникшей ошибки, а в переменную *ErrorIDEx* будет сохранен ответ с кодом ошибки от сервера FTP.

Настройка дополнительных параметров команды для сервера FTP

С помощью переменной *ExecOption* для команды загрузки файлов с сервера FTP можно задать ряд дополнительных параметров, определяющих особенности ее выполнения.

Для переменной *ExecOption* используется структурный тип данных _sFTP_EXEC_OPTION. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
ExecOption	Параметры выполнения команды для FTP	Дополнительные параметры выполнения команды для FTP	_sFTP_EXEC_OPTION	---	---	---
PassiveMode	Выбор пассивного режима	ИСТИНА: пассивный режим ЛОЖЬ: активный режим	BOOL	Зависит от типа данных.	---	ЛОЖЬ
ASCIIMode	Выбор режима ASCII	ИСТИНА: режим ASCII ЛОЖЬ: двоичный режим	BOOL			
FileRemove	Выбор удаления файлов после передачи *1	ИСТИНА: удалять файлы после передачи. ЛОЖЬ: не удалять файлы после передачи.	BOOL			
OverWrite	Выбор перезаписи	ИСТИНА: перезаписывать файлы в папке назначения. ЛОЖЬ: не перезаписывать файлы в папке назначения.	BOOL			
Reserved	Резерв	Зарезервировано для системы.	ARRAY[0..7] OF BYTE	---		0

*1. В случае сбоя передачи исходные передаваемые файлы не удаляются.

● PassiveMode (выбор пассивного режима)

Параметр PassiveMode (выбор пассивного режима) указывает, должен ли использоваться пассивный режим для запроса на установление соединения с сервером FTP с целью передачи данных.

Если пассивный режим не указан, для запроса на установление соединения с сервером FTP с целью передачи данных используется активный режим.

Дополнительные сведения о способах запроса на установление соединения см. в документе *Серия NJ/NX, модули ЦПУ — Встроенный порт EtherNet/IP. Руководство пользователя (Cat. No. W506)*.

В таблице ниже приведены возможные значения параметра *PassiveMode* и указано, что они означают.

Заданное значение	Значение
ИСТИНА	Запрос на подключение к серверу FTP для передачи данных выполняется в пассивном режиме. Запрос на подключение с целью передачи данных выполняется с клиента FTP.
ЛОЖЬ	Запрос на подключение к серверу FTP для передачи данных выполняется в активном режиме. Запрос на подключение с целью передачи данных выполняется с сервера FTP.

● ASCIIMode (выбор режима ASCII)

Параметр ASCIIMode (выбор режима ASCII) указывает, какой режим передачи должен использоваться для передачи данных из исходной системы в систему назначения: режим ASCII или двоичный режим.
Если режим ASCII не указан, в качестве режима передачи данных из исходной системы в систему назначения используется двоичный режим.
В таблице ниже приведены возможные значения параметра *ASCIIMode* и указано, что они означают.

Заданное значение	Значение
ИСТИНА	В качестве режима передачи данных из исходной системы в систему назначения используется режим ASCII. Коды перевода строки, используемые в исходной системе, преобразуются в соответствующие коды перевода строки для системы назначения.
ЛОЖЬ	В качестве режима передачи данных из исходной системы в систему назначения используется двоичный режим. Коды перевода строки передаются из исходной системы без преобразования.

● FileRemove (выбор удаления файлов после передачи)

Параметр FileRemove (выбор удаления файлов после передачи) указывает, должны ли удаляться файлы в исходной системе после того, как они переданы в целевую систему.
В таблице ниже приведены возможные значения параметра *FileRemove* и указано, что они означают.

Заданное значение	Значение
ИСТИНА	Исходные файлы после передачи удаляются.
ЛОЖЬ	Исходные файлы после передачи не удаляются.

● OverWrite (выбор перезаписи)

Параметр OverWrite (выбор перезаписи) указывает, должны ли перезаписываться файлы в папке назначения, если их имена совпадают с именами загружаемых файлов.
Если перезапись не указана, файлы, имена которых совпадают с именами файлов в папке назначения, не передаются.
Имена файлов не чувствительны к регистру.
В таблице ниже приведены возможные значения параметра *OverWrite* и указано, что они означают.

Заданное значение	Значение
ИСТИНА	Файлы в папке назначения перезаписываются.
ЛОЖЬ	Файлы в папке назначения не перезаписываются. Файлы не передаются в папку назначения.

Настройка выполнения повторных попыток соединения с сервером FTP

Можно настроить выполнение повторных попыток соединения с сервером FTP.
Выполнение повторных попыток настраивается точно так же, как и для команды FTPGetFileList.
См. *Настройка выполнения повторных попыток соединения с сервером FTP* на стр. 2-1325 для команды FTPGetFileList.

Отмена выполнения команды

Начавшееся выполнение команды FTPGetFile можно отменить.
При этом в переменные *GetNum* и GetFileResult[] будут сохранены имеющиеся на данный момент результаты загрузки файлов с сервера FTP.
Операция отмены производится точно так же, как и для команды FTPGetFileList. См. *Отмена выполнения команды* на стр. 2-1327 для команды FTPGetFileList.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta *1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta *2 _EIP2_EtnOnlineSta *3			
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Эта переменная показывает, распознана ли карта памяти SD и может ли она использоваться. ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если количество результатов загрузки файлов превышает количество элементов в массиве GetFileResult[], не помещающиеся результаты не сохраняются. Выход *Error* в этом случае в состояние ИСТИНА не переходит.
- Если длина имени файла превышает 255 символов, в элемент *Name* структуры GetFileResult[] сохраняются только первые 255 символов. Выход *Error* в этом случае в состояние ИСТИНА не переходит.
- Допускается одновременное выполнение не более 3 экземпляров следующих команд: FTPGetFileList, FTPGetFile, FTPPutFile, FTPRemoveFile и FTPRemoveDir.
- Если при указании имени файла использован подстановочный знак и возникла ошибка более чем для одного файла, в переменные *ErrorID* и *ErrorIDEx* сохраняются результаты первого файла, для которого значение флага ошибки *GetFileResult[].TxError* = ИСТИНА, из всех файлов, для которых сохранены результаты в переменную GetFileResult[].

- Имена файлов не чувствительны к регистру. Следовательно, если имена файлов в папке назначения и имена передаваемых файлов различаются только регистром букв, считается, что имена файлов совпадают. В этом случае происходит следующее:

Значение *OverWrite*	Выбор перезаписи	Обработка
ИСТИНА	Перезаписывать	Файлы перезаписываются.
ЛОЖЬ	Не перезаписывать	Файлы в папке назначения не перезаписываются. Файлы не передаются в папку назначения.

- Если файла с указанным именем *FileName* не существует в указанном каталоге на сервере FTP, происходит ошибка передачи и значение флага ошибки *GetFileResult[].TxError* меняется на ИСТИНА.
- Если указанное в параметре *FileName* имя является именем каталога, происходит ошибка передачи и значение флага ошибки *GetFileResult[].TxError* меняется на ИСТИНА.
- Если параметр *ExecOption.FileRemove* = ИСТИНА, а для файла с указанным именем *FileName* установлен атрибут «Только чтение», возникает ошибка удаления и значение флага ошибки *GetFileResult[].RemoveError* меняется на ИСТИНА.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение какого-либо входного параметра находится за пределами допустимого диапазона.
  b) ".." указано для уровня каталога в *SvrDirName* или *LocalDirName*.
  c) Для *SvrDirName* или *LocalDirName* указан неверный путь, например "//".
  d) На сервере FTP отсутствует каталог с указанным именем *SvrDirName*.
  e) Каталог с указанным именем *SvrDirName* на сервере FTP содержит больше 1000 файлов, которые можно загрузить.
  f) Файла с указанным именем *FileName* не существует в каталоге на сервере FTP, из которого нужно загрузить файлы.
  g) *ExecOption.OverWrite* = ЛОЖЬ, а в папке с указанным именем *SvrDirName* уже есть файл с таким же именем, что указано в параметре *FileName*.
  h) *ExecOption.FileRemove* = ИСТИНА, но для файла с именем, которое совпадает со значением в *FileName*, установлен атрибут «Только для чтения».
  i) Сервер FTP, указанный параметром *ConnectSvr*, отсутствует в сети или не работает.
  j) Не удалось получить доступ к файлу с указанным именем *FileName* из-за того, что отсутствуют права доступа к файлу или файл поврежден.
  k) Было выполнено более 3 следующих команд одновременно: FTPGetFileList, FTPGetFile, FTPPutFile, FTPRemoveFile и FTPRemoveDir.
  l) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  m) Карта памяти SD защищена от записи.
  n) На карте памяти SD недостаточно свободного места.
  o) Превышено максимальное количество файлов или каталогов на карте памяти SD.
- Для этой команды дополнительный код ошибки *ErrorIDEx* содержит код ответа FTP, который был возвращен сервером FTP. В следующей таблице перечислены типичные значения дополнительного кода ошибки *ErrorIDEx*, описано значение ошибок и приведены возможные способы их устранения. Дополнительные сведения см. в документации по серверу FTP. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = WORD#16#2407.

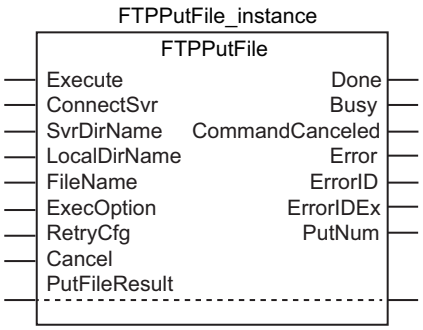
Значение *ErrorIDEx*	Значение	Способ устранения
16#000001A9	Не удалось установить соединение для передачи данных.	В случае использования протокола FTP для обмена данными с сервером FTP через Интернет проследите, чтобы для параметра Open mode (Режим открытия) для FTP не было установлено значение Active (Активный).
16#000001AA	Соединение было закрыто. Передача данных была прервана.	Проверьте соединение с сервером FTP. Убедитесь, что сервер FTP работает.
16#000001C2	Не удалось выполнить запрошенную операцию с файлом. Использовать файл было невозможно, например, из-за того, что он уже был открыт.	Убедитесь, что файл не открыт для какого-нибудь другого приложения.
16#00000212	Вход пользователя на сервер был невозможен.	Проверьте имя пользователя и пароль для входа на сервер FTP.
16#00000214	Для сохранения файлов требуется учетная запись.	Проверьте наличие у пользователя прав доступа к FTP.
16#00000226	Выполнить запрошенную операцию с файлом было невозможно, так как было невозможно использовать файл (например, не удалось получить доступ к файлу из-за того, что файл не был найден).	Убедитесь, что файл с указанным именем имеется в соответствующем каталоге на сервере FTP. Проверьте права доступа к указанному файлу.
16#00000229	Выполнение было невозможно, так как имя файла было неверным.	Проверьте права доступа к указанному каталогу.

Пример программы

См. *Пример программы* на стр. 2-1331 для команды FTPGetFileList.

FTPPutFile

Команда FTPPutFile загружает файл на сервер FTP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FTPPutFile	Поместить файл на сервер FTP	FB		FTPPutFile_instance(Execute, ConnectSvr, SvrDirName, LocalDirName, FileName, ExecOption, RetryCfg, Cancel, PutFileResult, Done, Busy, CommandCanceled, Error, ErrorID, ErrorIDEx, PutNum);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
ConnectSvr	Параметры подключенного сервера FTP	Вход	Параметры подключенного сервера FTP	---	---	*1
SvrDirName	Имя каталога сервера FTP		Имя каталога на сервере FTP, в который нужно загрузить файл	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)*2		"*3
LocalDirName	Имя локального каталога		Имя каталога, в котором хранится файл, загружаемый на сервер FTP	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)		«/»
FileName	Имя файла		Имя загружаемого файла*4			*1
ExecOption	Параметры выполнения команды для FTP		Дополнительные параметры выполнения команды для FTP	---		---
RetryCfg	Настройка повтора выполнения		Настройка повторных попыток выполнения команды			
Cancel	Отмена		ИСТИНА: выполнение команды отменено. ЛОЖЬ: выполнение команды не отменено.	Зависит от типа данных.		ЛОЖЬ
PutFile Result[] (массив)*5*6*7	Результаты загрузки файла	Вход-выход	Результаты загрузки файла	---	---	*1
CommandCanceled	Отмена завершена	Выход	ИСТИНА: отмена завершена. ЛОЖЬ: отмена не завершена.	Зависит от типа данных.	---	---
PutNum	Количество загружаемых файлов		Количество загружаемых файлов	---		

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.
*2. В именах каталогов сервера FTP нельзя использовать следующие символы:
* ? < > | "
*3. При входе на сервер FTP по умолчанию используется домашний каталог.
*4. В именах файлов допускается использовать подстановочные знаки.
*5. Массив может содержать не более 1000 элементов.
*6. Это одномерный массив. Если будет указан массив с двумя или большим числом измерений, произойдет ошибка сборки.
*7. Номер первого элемента массива: 0. Если для первого элемента массива будет указано число, отличное от 0, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ConnectSvr	Подробные сведения о структуре _sFTP_CONNECT_SVR см. в разделе *Функция* на стр. 2-1349.																			
SvrDirName																				ОК
LocalDirName																				ОК
FileName																				ОК
ExecOption	Подробные сведения о структуре _sFTP_EXEC_OPTION см. в разделе *Настройка дополнительных параметров команды для сервера FTP* на стр. 2-1351.																			
RetryCfg	Подробные сведения о структуре _sFTP_RETRY_CFG см. в разделе *Настройка выполнения повторных попыток соединения с сервером FTP* на стр. 2-1343.																			
Cancel	ОК																			
PutFileResult[] (массив)	Подробные сведения о структуре _sFTP_FILE_RESULT см. в разделе *Функция* на стр. 2-1349.																			
CommandCanceled	ОК																			
PutNum							ОК													

Функция

Команда FTPPutFile загружает файл с указанным именем (*FileName*) из указанного каталога (*LocalDirName*) на карте памяти SD в указанный каталог (*SvrDirName*) на подключенном сервере FTP (*ConnectSvr*).
Если каталога с указанным именем *SvrDirName* на сервере FTP не существует, создается новый каталог и указанный файл загружается в него.
В переменной *FileName* допускается использовать подстановочные знаки. Это позволяет передать более одного файла за раз.

Результаты загрузки для каждого файла сохраняются в переменную PutFileResult[].
Количество файлов, которые нужно загрузить, указывается в переменной *PutNum*.
В случае использования подстановочного знака в параметре *FileName* следует указать количество всех файлов, имена которых удовлетворяют подстановочному знаку.

Если фактическое количество переданных файлов будет другим, значение переменной *PutFileResult[].TxError* поменяется на ИСТИНА.
Если при удалении исходного файла после его загрузки возникнет ошибка, значение переменной *PutFileResult[].RemoveError* поменяется на ИСТИНА.

Для переменной *ConnectSvr* используется структурный тип данных _sFTP_CONNECT_SVR.
Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
ConnectSvr	Параметры подключенного сервера FTP	Параметры подключенного сервера FTP	_sFTP_CONNECT_SVR	---	---	---
Adr	Адрес	IP-адрес или имя хоста*1	STRING	1...200 байт*2	---	---
PortNo	Номер порта	Номер порта TCP сервера FTP для соединения	UINT	0...65535*3		
UserName	Имя пользователя	Имя пользователя на сервере FTP	STRING	Макс. 33 байт*4*5*6		
Password	Пароль	Пароль для входа на сервер FTP	STRING	Макс. 33 байт*4*5*6		

*1. Чтобы можно было указать имя хоста, должны быть отдельно настроены параметры DNS или хостов.
*2. Можно использовать следующие однобайтовые символы: "A...Z", "a...z", "0...9", "-" (дефис), "." (точка) и "_" (подчеркивание).
*3. Если указано значение 0, используется порт TCP с номером 21.
*4. Можно использовать следующие однобайтовые символы: "A...Z", "a...z", "0...9", "-" (дефис), "." (точка) и "_" (подчеркивание). Для модуля ЦПУ с версией модуля 1.16 или более поздней также можно использовать символы "\" (обратная косая черта) и "@".
*5. При подсчете количества байтов также следует учитывать нулевой символ (NULL).
*6. В случае модулей ЦПУ с версией модуля 1.08 следует указать текстовую строку, состоящую из одного или более символов. Если будет указана текстовая строка, содержащая только завершающий пустой символ (NULL), произойдет ошибка.

Для переменной PutFileResult[] используется структурный тип данных _sFTP_FILE_RESULT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
PutFileResult	Результаты загрузки файла	Результаты передачи файла	_sFTP_FILE_RESULT	---	---	---
Name	Имя файла*1	Имя переданного файла	STRING	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	---
TxError	Ошибка передачи	ИСТИНА: передача завершилась с ошибкой. ЛОЖЬ: передача завершилась нормально.	BOOL	Зависит от типа данных.		
RemoveError	Ошибка удаления	ИСТИНА: удаление завершилось с ошибкой. ЛОЖЬ: удаление завершилось нормально.	BOOL			
Reserved	Резерв	Зарезервировано для системы.	ARRAY[0..3] OF BYTE	---		0

*1. Включает расширение имени файла.

Использование подстановочных знаков при указании имен файлов

При указании имен загружаемых файлов допускается использовать подстановочные знаки. Применяются те же правила использования подстановочных знаков, что и для команды FTPGetFile. См. *Использование подстановочных знаков при указании имен файлов* на стр. 2-1341 для команды FTPGetFile.

Настройка дополнительных параметров команды для сервера FTP

Для команды загрузки файлов на сервер FTP можно задать дополнительные параметры выполнения.
Предусмотрены те же параметры, что и для команды FTPGetFile. См. *Настройка дополнительных параметров команды для сервера FTP* на стр. 2-1341 для команды FTPGetFile.

Настройка выполнения повторных попыток соединения с сервером FTP

Можно настроить выполнение повторных попыток соединения с сервером FTP.
Выполнение повторных попыток настраивается точно так же, как и для команды FTPGetFileList. См. *Настройка выполнения повторных попыток соединения с сервером FTP* на стр. 2-1325 для команды FTPGetFileList.

Отмена выполнения команды

Начавшееся выполнение команды FTPPutFile можно отменить.
При этом в переменные *PutNum* и PutFileResult[] будут сохранены имеющиеся на данный момент результаты загрузки файлов на сервер FTP.
Операция отмены производится точно так же, как и для команды FTPGetFileList. См. *Отмена выполнения команды* на стр. 2-1327 для команды FTPGetFileList.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta *1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta *2 _EIP2_EtnOnlineSta *3			
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Эта переменная показывает, распознана ли карта памяти SD и может ли она использоваться. ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если количество результатов загрузки файлов превышает количество элементов в массиве PutFileResult[], не помещающиеся результаты не сохраняются. Выход *Error* в этом случае в состояние ИСТИНА не переходит.
- Если длина имени файла превышает 255 символов, в элемент *Name* структуры PutFileResult[] сохраняются только первые 255 символов. Выход *Error* в этом случае в состояние ИСТИНА не переходит.
- Допускается одновременное выполнение не более 3 экземпляров следующих команд: FTPGetFileList, FTPGetFile, FTPPutFile, FTPRemoveFile и FTPRemoveDir.
- Если при указании имени файла использован подстановочный знак и возникла ошибка более чем для одного файла, в переменные *ErrorID* и *ErrorIDEx* сохраняются результаты первого файла, для которого значение флага ошибки *PutFileResult[].TxError* = ИСТИНА, из всех файлов, для которых сохранены результаты в переменную PutFileResult[].
- Имена файлов не чувствительны к регистру. Следовательно, если имена файлов в папке назначения и имена передаваемых файлов различаются только регистром букв, считается, что имена файлов совпадают. В этом случае происходит следующее:

Значение *OverWrite*	Выбор перезаписи	Обработка
ИСТИНА	Перезаписывать	Если перезапись не указана, работа команды зависит от особенностей сервера FTP.
ЛОЖЬ	Не перезаписывать	Файлы в папке назначения не перезаписываются. Файлы не передаются в папку назначения.

- Если файла с указанным именем *FileName* не существует в указанном каталоге на карте памяти SD, происходит ошибка передачи и значение флага ошибки *PutFileResult[].TxError* меняется на ИСТИНА.
- Если указанное в параметре *FileName* имя является именем каталога, происходит ошибка передачи и значение флага ошибки *PutFileResult[].TxError* меняется на ИСТИНА.
- Если параметр *ExecOption.FileRemove* = ИСТИНА, а для файла с указанным именем *FileName* установлен атрибут «Только чтение», возникает ошибка удаления и значение флага ошибки *PutFileResult[].RemoveError* меняется на ИСТИНА.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение какого-либо входного параметра находится за пределами допустимого диапазона.
  b) ".." указано для уровня каталога в *SvrDirName* или *LocalDirName*.
  c) Для *SvrDirName* или *LocalDirName* указан неверный путь, например "//".
  d) На сервере FTP отсутствует каталог с указанным именем *SvrDirName*.
  e) На клиенте FTP отсутствует каталог с указанным именем *LocalDirName*.
  f) Каталог с указанным именем *LocalDirName* содержит больше 1000 файлов, которые можно загрузить.

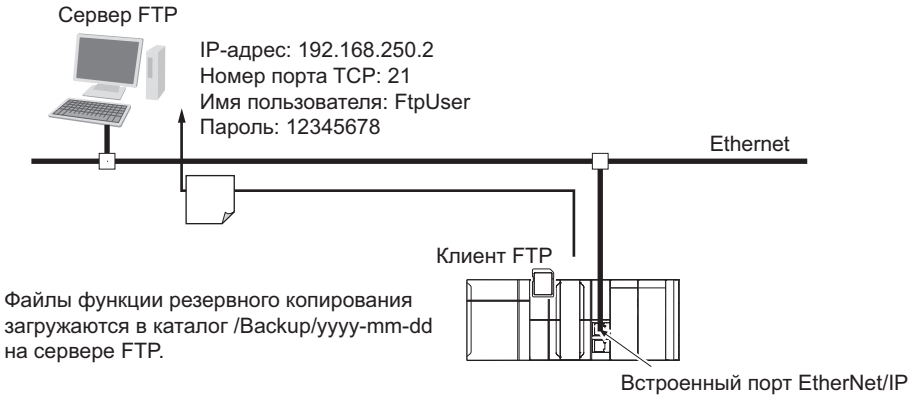
g) Файла с указанным именем *FileName* не существует в каталоге на карте памяти SD, из которого нужно загрузить файлы.
h) *ExecOption.OverWrite* = ЛОЖЬ, а в папке с указанным именем *SvrDirName* уже есть файл с таким же именем, что указано в параметре *FileName*.
i) *ExecOption.FileRemove* = ИСТИНА, но для файла с именем, которое совпадает со значением в *FileName*, установлен атрибут «Только для чтения».
j) Сервер FTP, указанный параметром *ConnectSvr*, отсутствует в сети или не работает.
k) Не удалось получить доступ к файлу с указанным именем *FileName* из-за того, что отсутствуют права доступа к файлу или файл поврежден.
l) Было выполнено более 3 следующих команд одновременно: FTPGetFileList, FTPGetFile, FTPPutFile, FTPRemoveFile и FTPRemoveDir.
m) Карта памяти SD не находится в состоянии, в котором возможно ее использование.

- Для этой команды дополнительный код ошибки *ErrorIDEx* содержит код ответа FTP, который был возвращен сервером FTP. В следующей таблице перечислены типичные значения дополнительного кода ошибки *ErrorIDEx*, описано значение ошибок и приведены возможные способы их устранения. Дополнительные сведения см. в документации по серверу FTP. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = WORD#16#2407.

Значение *ErrorIDEx*	Значение	Способ устранения
16#000001A9	Не удалось установить соединение для передачи данных.	В случае использования протокола FTP для обмена данными с сервером FTP через Интернет проследите, чтобы для параметра Open mode (Режим открытия) для FTP не было установлено значение Active (Активный).
16#000001AA	Соединение было закрыто. Передача данных была прервана.	Проверьте соединение с сервером FTP. Убедитесь, что сервер FTP работает.
16#000001C2	Не удалось выполнить запрошенную операцию с файлом. Использовать файл было невозможно, например, из-за того, что он уже был открыт.	Убедитесь, что файл не открыт для какого-нибудь другого приложения.
16#00000212	Вход пользователя на сервер был невозможен.	Проверьте имя пользователя и пароль для входа на сервер FTP.
16#00000214	Для сохранения файлов требуется учетная запись.	Проверьте наличие у пользователя прав доступа к FTP.
16#00000226	Выполнить запрошенную операцию с файлом было невозможно, так как было невозможно использовать файл (например, не удалось получить доступ к файлу из-за того, что файл не был найден).	Убедитесь, что файл с указанным именем имеется в соответствующем каталоге на сервере FTP. Проверьте права доступа к указанному файлу.
16#00000229	Выполнение было невозможно, так как имя файла было неверным.	Проверьте права доступа к указанному каталогу.

Пример программы

В качестве примера рассмотрим программу, которая выполняет резервное копирование данных на карту памяти SD, а затем загружает все файлы, относящиеся к резервной копии, в каталог «/Backup/yyyy-mm-dd» на сервере FTP.



Контроллер подключен к серверу FTP по сети EtherNet/IP. Значения параметров для подключения к серверу FTP приведены в следующей таблице.

Параметр	Значение
IP-адрес	192.168.250.2
Номер порта TCP	21
Имя пользователя	FtpUser
Пароль	12345678

Соблюдается приведенный ниже порядок действий.

1 Используется команда BackupToMemoryCard для сохранения файлов резервной копии данных контроллера в корневой каталог на карте памяти SD.

2 С помощью команды FTPPutFile файлы, относящиеся к резервной копии, передаются в каталог «/Backup/yyyy-mm-dd» на сервере FTP. При указании имен передаваемых файлов используется подстановочный знак «*.*».

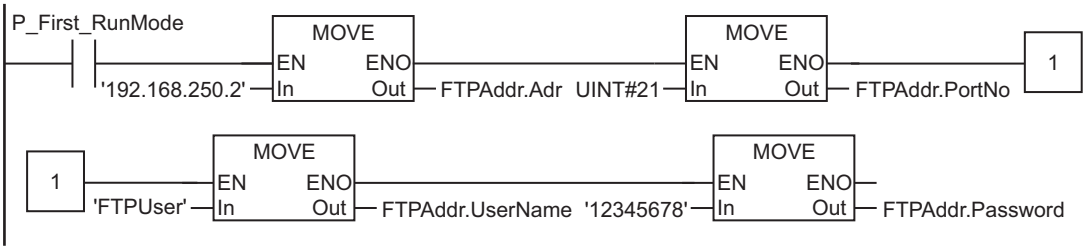
3 Если все операции завершаются нормально, выполняется обработка нормального завершения.
А если возникает ошибка, то выполняется обработка для завершения с ошибкой.

Программа на языке LD

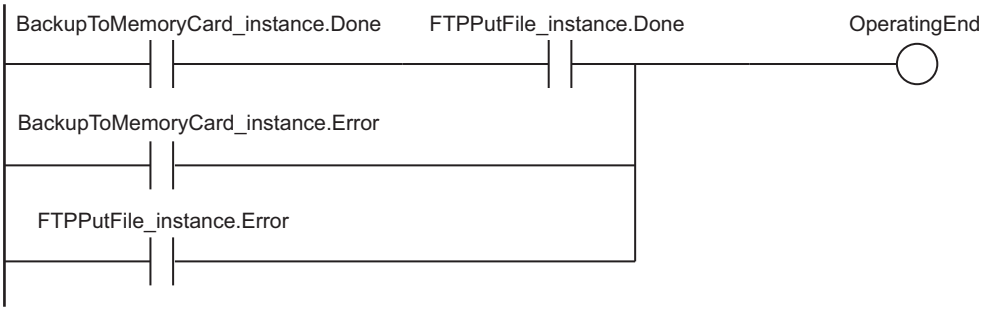
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	FTPPutFile_instance	FTPPutFile		Экземпляр команды FTPPutFile
	FTPAddr	_sFTP_CONNECT_SVR	(Adr := '', PortNo := 0, UserName := '', Password := '')	Параметры подключенного сервера FTP
	PutResult	ARRAY[0..0] OF _sFTP_FILE_RESULT	[(Name := '', TxError := False, RemoveError := False, Reserved := [4(16#0)])]	Результаты загрузки файла
	RS_instance	RS		Экземпляр команды RS
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена
	Trigger	BOOL	ЛОЖЬ	Условие выполнения

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Operating	BOOL	ЛОЖЬ	Обработка
	BackupToMemoryCard_instance	BackupToMemoryCard		Экземпляр команды BackupToMemoryCard

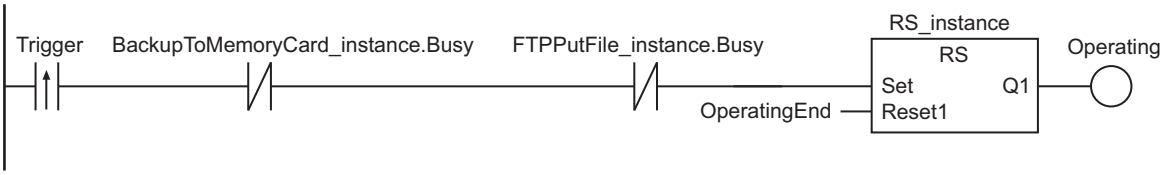
Подготовка параметров подключенного сервера FTP.



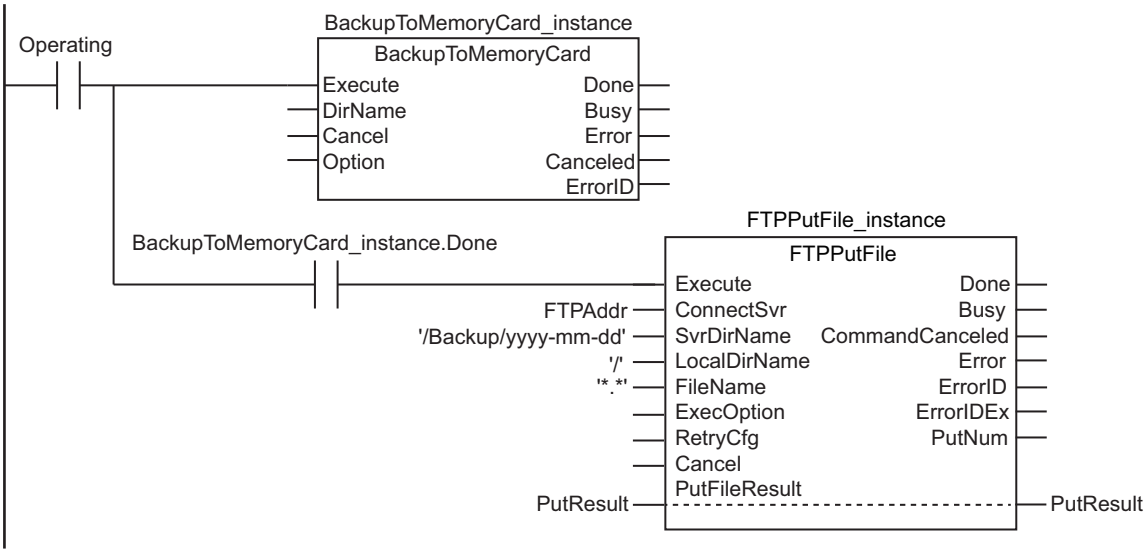
Определение, завершено ли выполнение команды.

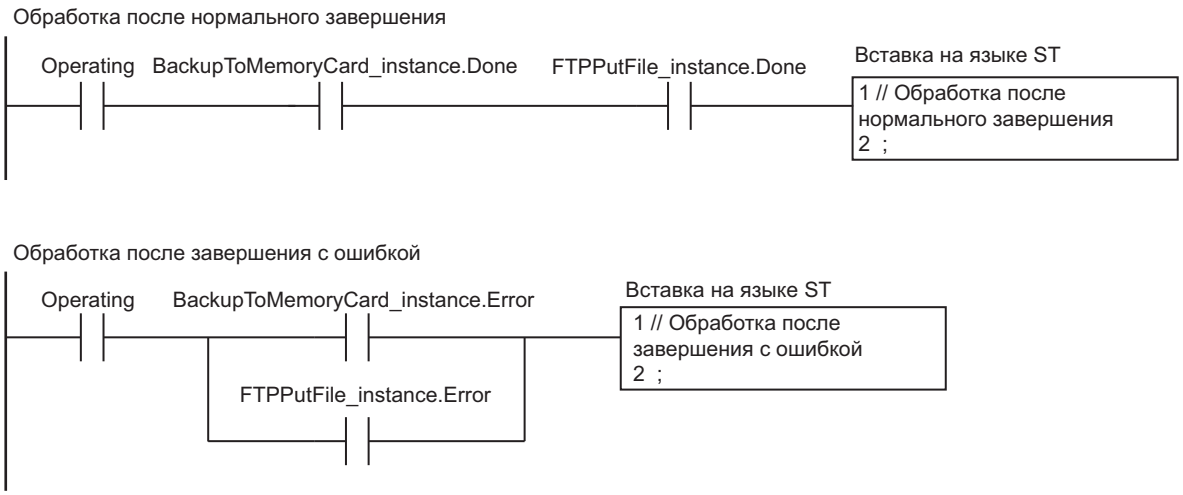


Прием условия выполнения.



Выполнение команд BackupToMemoryCard и FTPPutFile.





Программа на языке ST

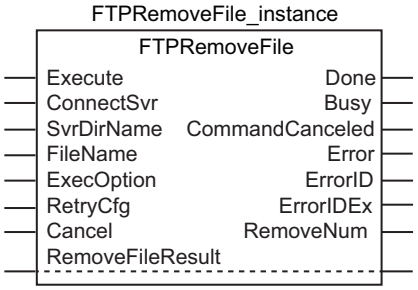
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	R_TRIG_instance	R_TRIG		Экземпляр команды R_TRIG
	UP_Q	BOOL	ЛОЖЬ	Выход условия запуска
	FTPPutFile_instance	FTPPutFile		Экземпляр команды FTPPutFile
	DoFTPTrigger	BOOL	ЛОЖЬ	Условие выполнения для команд BackupToMemoryCard и FTPPutFile
	FTPAddr	_sFTP_CONNECT_SVR	(Adr := '', PortNo := 0, UserName := '', Password := '')	Параметры подключенного сервера FTP
	PutResult	ARRAY[0..0] OF _sFTP_FILE_RESULT	[(Name := '', TxError := False, RemoveError := False, Reserved := [4(16#0)])]	Результаты загрузки файла
	Stage	UINT	0	Этап выполнения команды
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	BackupToMemoryCard_instance	BackupToMemoryCard		Экземпляр команды BackupToMemoryCard

```
// Подготовка параметров подключенного сервера FTP.
IF P_First_RunMode THEN
   FTPAddr.Adr      := '192.168.250.2';// Адрес
   FTPAddr.PortNo   := UINT#21;        // Номер порта
   FTPAddr.UserName := 'FtpUser';      // Имя пользователя
   FTPAddr.Password := '12345678';     // Пароль
END_IF;

// Прием условия выполнения.
R_TRIG_instance(Trigger, UP_Q);
```

```
IF ( (UP_Q = TRUE) AND (BackupToMemoryCard_instance.Busy = FALSE) AND
     (FTPPutFile_instance.Busy = FALSE) ) THEN
   DoFTPTrigger := TRUE;
   Stage := INT#1;
   BackupToMemoryCard_instance(  // Инициализация экземпляра.
      Execute := FALSE) ;
   FTPPutFile_instance(  // Инициализация экземпляра.
      Execute       := FALSE,
      ConnectSvr    := FTPAddr,
      SvrDirName    := '/Backup/yyyy-mm-dd',
      LocalDirName  := '/',
      FileName      := '*.*',
      PutFileResult := PutResult) ;
END_IF;

IF (DoFTPTrigger = TRUE) THEN
   CASE Stage OF
      1 :// Выполнение команды BackupToMemoryCard.
         BackupToMemoryCard_instance(
            Execute := TRUE) ;// Выполнение
         IF (BackupToMemoryCard_instance.Done = TRUE) THEN
            Stage := INT#2;  // К следующему этапу
         ELSIF (BackupToMemoryCard_instance.Error = TRUE) THEN
            Stage := INT#10; // Завершение с ошибкой
         END_IF;
      2 : // Выполнение команды FTPPutFile.
         FTPPutFile_instance(
            Execute       := TRUE,                // Выполнение
            ConnectSvr    := FTPAddr,             // Подключенный FTP-сервер
            SvrDirName    := '/Backup/yyyy-mm-dd',// Имя каталога на FTP-сервере
            LocalDirName  := '/',                 // Имя локального каталога
            FileName      := '*.*',               // Имя файла
            PutFileResult := PutResult) ;         // Результаты загрузки файлов
         IF (FTPPutFile_instance.Done = TRUE) THEN
            Stage := INT#0;  // Нормальное завершение
         ELSIF (FTPPutFile_instance.Error = TRUE) THEN
            Stage := INT#20; // Завершение с ошибкой
         END_IF;
      0 :  // Обработка после нормального завершения
         DoFTPTrigger:=FALSE;
         Trigger     :=FALSE;
      ELSE // Обработка после завершения с ошибкой
         DoFTPTrigger:=FALSE;
         Trigger     :=FALSE;
   END_CASE;
END_IF;
```

FTPRemoveFile

Команда FTPRemoveFile удаляет файл с сервера FTP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FTPRemoveFile	Удаление файла с сервера FTP	FB		FTPRemoveFile_instance(Execute, ConnectSvr, SvrDirName, FileName, ExecOption, RetryCfg, Cancel, RemoveFileResult, Done, Busy, CommandCanceled, Error, ErrorID, ErrorIDEx, RemoveNum);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
ConnectSvr	Параметры подключенного сервера FTP	Вход	Параметры подключенного сервера FTP	---	---	*1
SvrDirName	Имя каталога сервера FTP		Имя каталога на сервере FTP, содержащего удаляемый файл	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)*2		""*3
FileName	Имя файла		Имя удаляемого файла*4	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)*5		*1
ExecOption	Параметры выполнения команды для FTP		Дополнительные параметры выполнения команды для FTP	---		---
RetryCfg	Настройка повтора выполнения		Настройка повторных попыток выполнения команды			
Cancel	Отмена		ИСТИНА: выполнение команды отменено. ЛОЖЬ: выполнение команды не отменено.	Зависит от типа данных.		
RemoveFileResult[] (массив)*6*7*8	Результаты удаления файла	Вход-выход	Результаты удаления файла	---	---	*1
CommandCanceled	Отмена завершена	Выход	ИСТИНА: отмена завершена. ЛОЖЬ: отмена не завершена.	Зависит от типа данных.	---	---
RemoveNum	Количество удаляемых файлов		Количество файлов, которые нужно удалить	---		

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.
*2. В именах каталогов сервера FTP нельзя использовать следующие символы:
* ? < > | "
*3. При входе на сервер FTP по умолчанию используется домашний каталог.
*4. В именах файлов допускается использовать подстановочные знаки.
*5. В именах файлов нельзя использовать следующий символ:
|
*6. Массив может содержать не более 1000 элементов.
*7. Это одномерный массив. Если будет указан массив с двумя или большим числом измерений, произойдет ошибка сборки.
*8. Номер первого элемента массива: 0. Если для первого элемента массива будет указано число, отличное от 0, произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ConnectSvr	Подробные сведения о структуре _sFTP_CONNECT_SVR см. в разделе *Функция* на стр. 2-1360.																			
SvrDirName																				ОК
FileName																				ОК
ExecOption	Подробные сведения о структуре _sFTP_EXEC_OPTION см. в разделе *Настройка дополнительных параметров команды для сервера FTP* на стр. 2-1362.																			
RetryCfg	Подробные сведения о структуре _sFTP_RETRY_CFG см. в разделе *Настройка выполнения повторных попыток соединения с сервером FTP* на стр. 2-1343.																			
Cancel	ОК																			
Remove FileResult[] (массив)	Подробные сведения о структуре _sFTP_FILE_RESULT см. в разделе *Функция* на стр. 2-1360.																			
CommandCanceled	ОК																			
RemoveNum							ОК													

Функция

Команда FTPRemoveFile удаляет указанный файл (*FileName*) в указанном каталоге (*SvrDirName*) на подключенном сервере FTP (*ConnectSvr*).
В переменной *FileName* допускается использовать подстановочные знаки. Это позволяет удалить более одного файла за раз.

Результаты удаления сохраняются в переменную RemoveFileResult[] отдельно для каждого файла. Количество файлов, которые нужно удалить, указывается в переменной *RemoveNum*.
В случае использования подстановочного знака в параметре *FileName* следует указать количество всех файлов, имена которых удовлетворяют подстановочному знаку.

Если фактическое количество удаленных файлов будет другим, значение переменной *RemoveFileResult[].RemoveError* поменяется на ИСТИНА.

Для переменной *ConnectSvr* используется структурный тип данных _sFTP_CONNECT_SVR. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
ConnectSvr	Параметры подключенного сервера FTP	Параметры подключенного сервера FTP	_sFTP_CONNECT_SVR	---	---	---
Adr	Адрес	IP-адрес или имя хоста*1	STRING	1...200 байт*2	---	---
PortNo	Номер порта	Номер порта TCP сервера FTP для соединения	UINT	0...65535*3		
UserName	Имя пользователя	Имя пользователя на сервере FTP	STRING	Макс. 33 байт*4*5*6		
Password	Пароль	Пароль для входа на сервер FTP	STRING	Макс. 33 байт*4*5*6		

*1. Чтобы можно было указать имя хоста, должны быть отдельно настроены параметры DNS или хостов.
*2. Можно использовать следующие однобайтовые символы: "A...Z", "a...z", "0...9", "-" (дефис), "." (точка) и "_" (подчеркивание).
*3. Если указано значение 0, используется порт TCP с номером 21.
*4. Можно использовать следующие однобайтовые символы: "A...Z", "a...z", "0...9", "-" (дефис), "." (точка) и "_" (подчеркивание). Для модуля ЦПУ с версией модуля 1.16 или более поздней также можно использовать символы "\" (обратная косая черта) и "@".
*5. При подсчете количества байтов также следует учитывать нулевой символ (NULL).
*6. В случае модулей ЦПУ с версией модуля 1.08 следует указать текстовую строку, состоящую из одного или более символов. Если будет указана текстовая строка, содержащая только завершающий пустой символ (NULL), произойдет ошибка.

Для переменной RemoveFileResult[] используется структурный тип данных _sFTP_FILE_RESULT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
RemoveFileResult	Результаты удаления файла	Результаты передачи файла	_sFTP_FILE_RESULT	---	---	---
Name	Имя файла*1	Имя переданного файла	STRING	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	---
TxError	Ошибка передачи	ИСТИНА: передача завершилась с ошибкой. ЛОЖЬ: передача завершилась нормально.	BOOL	Зависит от типа данных.		
RemoveError	Ошибка удаления	ИСТИНА: удаление завершилось с ошибкой. ЛОЖЬ: удаление завершилось нормально.	BOOL			
Reserved	Резерв	Зарезервировано для системы.	ARRAY[0..3] OF BYTE	---		0

*1. Включает расширение имени файла.

Использование подстановочных знаков при указании имен файлов

При указании имен удаляемых файлов допускается использовать подстановочные знаки. Применяются те же правила использования подстановочных знаков, что и для команды FTPGetFile. См. *Использование подстановочных знаков при указании имен файлов* на стр. 2-1341 для команды FTPGetFile.

Настройка дополнительных параметров команды для сервера FTP

С помощью переменной *ExecOption* для команды удаления файлов с сервера FTP можно задать ряд дополнительных параметров, определяющих особенности ее выполнения. Предусмотрены те же параметры, что и для команды FTPGetFile. См. *Настройка дополнительных параметров команды для сервера FTP* на стр. 2-1341 для команды FTPGetFileList. Однако для этой команды действителен только параметр *ExecOption.PassiveMode*.

Настройка выполнения повторных попыток соединения с сервером FTP

Можно настроить выполнение повторных попыток соединения с сервером FTP. Выполнение повторных попыток настраивается точно так же, как и для команды FTPGetFileList. См. *Настройка выполнения повторных попыток соединения с сервером FTP* на стр. 2-1325 для команды FTPGetFileList.

Отмена выполнения команды

Начавшееся выполнение команды FTPRemoveFile можно отменить. При этом в переменные *RemoveNum* и RemoveFileResult[] будут сохранены имеющиеся на данный момент результаты удаления файлов с сервера FTP. Операция отмены производится точно так же, как и для команды FTPGetFileList. См. *Отмена выполнения команды* на стр. 2-1327 для команды FTPGetFileList.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.
*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

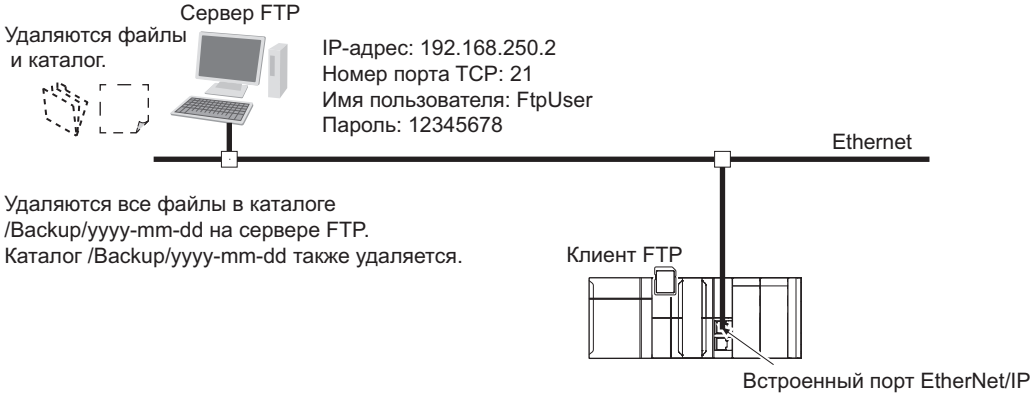
Меры предосторожности для обеспечения надлежащей эксплуатации

- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если количество удаленных файлов превышает количество элементов в массиве RemoveFileResult[], не помещающиеся результаты не сохраняются. Выход *Error* в этом случае в состояние ИСТИНА не переходит.
- Если длина имени файла превышает 255 символов, в элемент Name структуры *RemoveFileResult[].Name* сохраняются только первые 255 символов. Выход *Error* в этом случае в состояние ИСТИНА не переходит.
- Допускается одновременное выполнение не более 3 экземпляров следующих команд: FTPGetFileList, FTPGetFile, FTPPutFile, FTPRemoveFile и FTPRemoveDir.
- Если при указании имени файла использован подстановочный знак и возникла ошибка более чем для одного файла, в переменные *ErrorID* и *ErrorIDEx* сохраняются результаты первого файла, для которого значение флага ошибки *RemoveFileResult[].TxError* = ИСТИНА, из всех файлов, для которых сохранены результаты в переменную RemoveFileResult[].
- В указанных ниже случаях значение переменной *RemoveFileResult[].RemoveError* меняется на ИСТИНА.
  a) На сервере FTP отсутствует файл с указанным именем *FileName*.
  b) Для файла с указанным именем *FileName* установлен атрибут «Только чтение».
  c) Указанное в переменной *FileName* имя является именем каталога.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение какого-либо входного параметра находится за пределами допустимого диапазона.
  b) ".." указано для уровня каталога в *SvrDirName*.
  c) Для *SvrDirName* указан неверный путь, например "//".
  d) На сервере FTP отсутствует каталог с указанным именем *SvrDirName*.
  e) Каталог с указанным именем *SvrDirName* содержит больше 1000 файлов, которые можно удалить.
  f) В каталоге на сервере FTP нет файлов, чьи имена подходят для имени файла, указанного в переменной *FileName* с использованием подстановочных знаков.
  g) Для файла с указанным именем *FileName* установлен атрибут «Только чтение».
  h) Сервер FTP, указанный параметром *ConnectSvr*, отсутствует в сети или не работает.
  i) Было выполнено более 3 следующих команд одновременно: FTPGetFileList, FTPGetFile, FTPPutFile, FTPRemoveFile и FTPRemoveDir.
- Для этой команды дополнительный код ошибки *ErrorIDEx* содержит код ответа FTP, который был возвращен сервером FTP. В следующей таблице перечислены типичные значения дополнительного кода ошибки *ErrorIDEx*, описано значение ошибок и приведены возможные способы их устранения. Дополнительные сведения см. в документации по серверу FTP. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = WORD#16#2407.

Значение *ErrorIDEx*	Значение	Способ устранения
16#000001A9	Не удалось установить соединение для передачи данных.	В случае использования протокола FTP для обмена данными с сервером FTP через Интернет проследите, чтобы для параметра Open mode (Режим открытия) для FTP не было установлено значение Active (Активный).
16#000001AA	Соединение было закрыто. Передача данных была прервана.	Проверьте соединение с сервером FTP. Убедитесь, что сервер FTP работает.
16#000001C2	Не удалось выполнить запрошенную операцию с файлом. Использовать файл было невозможно, например, из-за того, что он уже был открыт.	Убедитесь, что файл не открыт для какого-нибудь другого приложения.
16#00000212	Вход пользователя на сервер был невозможен.	Проверьте имя пользователя и пароль для входа на сервер FTP.
16#00000214	Для сохранения файлов требуется учетная запись.	Проверьте наличие у пользователя прав доступа к FTP.
16#00000226	Выполнить запрошенную операцию с файлом было невозможно, так как было невозможно использовать файл (например, не удалось получить доступ к файлу из-за того, что файл не был найден).	Убедитесь, что файл с указанным именем имеется в соответствующем каталоге на сервере FTP. Проверьте права доступа к указанному файлу.
16#00000229	Выполнение было невозможно, так как имя файла было неверным.	Проверьте права доступа к указанному каталогу.

Пример программы

Данная программа удаляет все файлы в каталоге «/Backup/yyyy-mm-dd» на сервере FTP. После этого также удаляется и сам каталог «/Backup/yyyy-mm-dd».



Контроллер подключен к серверу FTP по сети EtherNet/IP. Значения параметров для подключения к серверу FTP приведены в следующей таблице.

Параметр	Значение
IP-адрес	192.168.250.2
Номер порта TCP	21
Имя пользователя	FtpUser
Пароль	12345678

Соблюдается приведенный ниже порядок действий.

1 Используется команда FTPRemoveFile для удаления всех файлов в каталоге «/Backup/yyyy-mm-dd» на сервере FTP. При указании имен удаляемых файлов используется подстановочный знак «*.*».

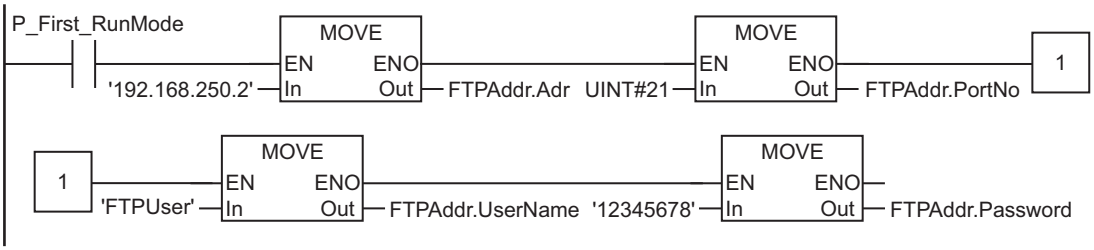
2 Используется команда FTPRemoveDir для удаления каталога «/Backup/yyyy-mm-dd» с сервера FTP.

3 Если все операции завершаются нормально, выполняется обработка нормального завершения. А если возникает ошибка, то выполняется обработка для завершения с ошибкой.

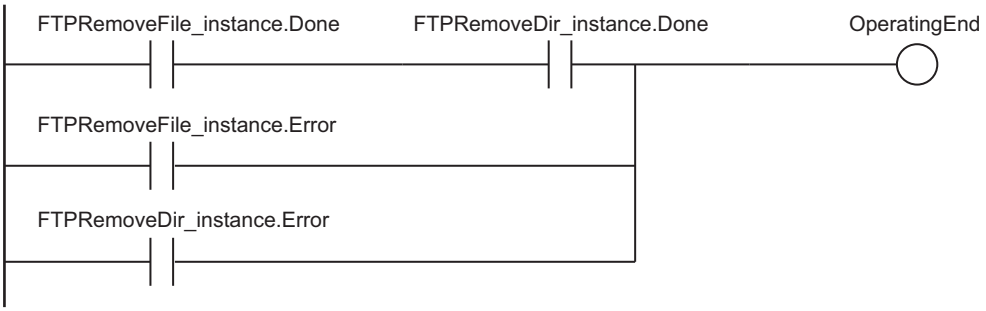
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	FTPRemoveFile_instance	FTPRemoveFile		Экземпляр команды FTPRemoveFile
	FTPRemoveDir_instance	FTPRemoveDir		Экземпляр команды FTPRemoveDir
	FTPAddr	_sFTP_CONNECT_SVR	(Adr := '', PortNo := 0, UserName := '', Password := '')	Параметры подключенного сервера FTP
	RemoveResult	ARRAY[0..0] OF _sFTP_FILE_RESULT	[(Name := '', TxError := False, RemoveError := False, Reserved := [4(16#0)])]	Результаты удаления файла
	RS_instance	RS		Экземпляр команды RS
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка

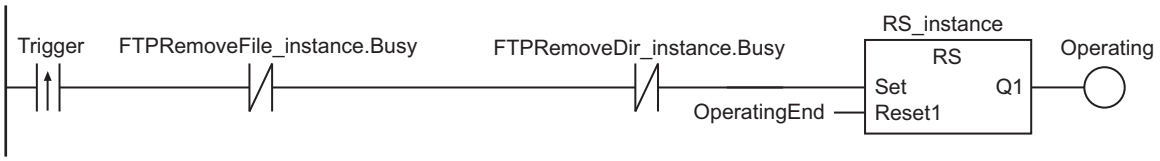
Подготовка параметров подключенного сервера FTP.



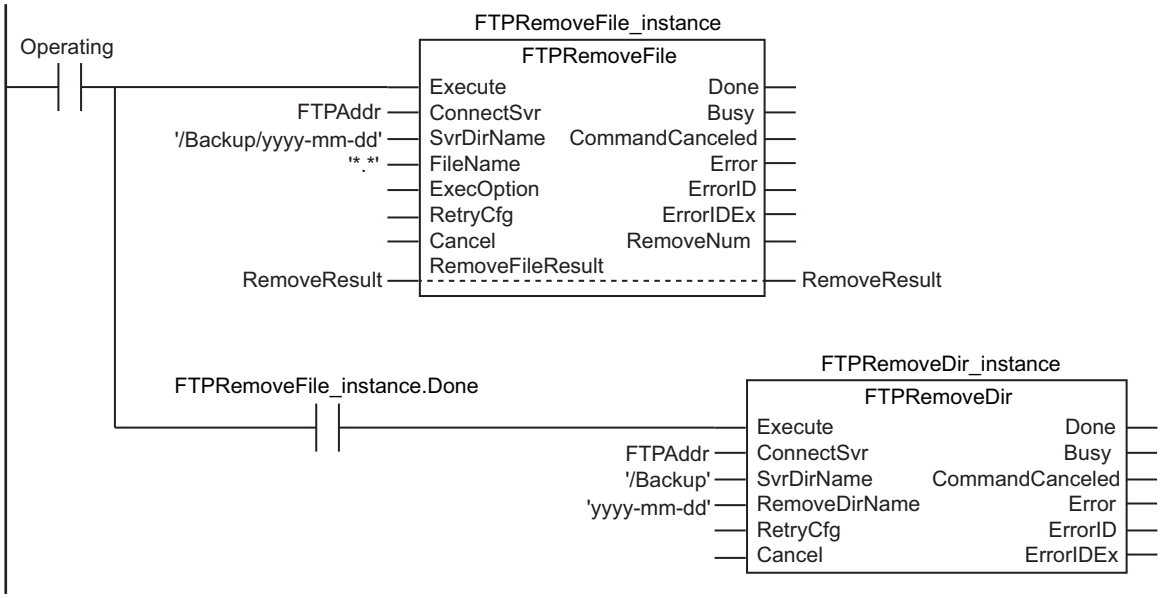
Определение, завершено ли выполнение команды.



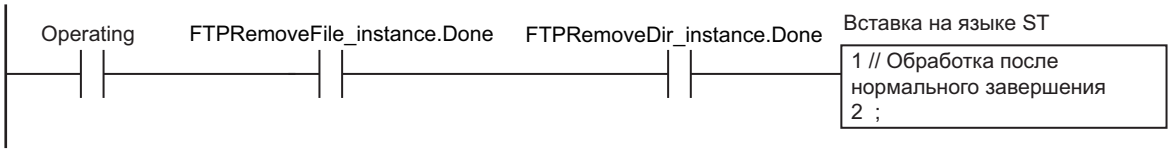
Прием условия выполнения.



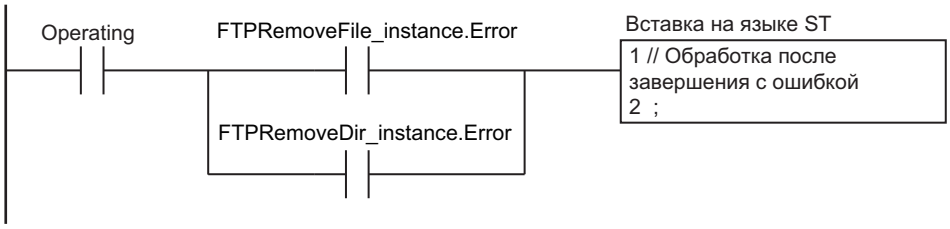
Выполнение команд FTPRemoveFile и FTPRemoveDir.



Обработка после нормального завершения



Обработка после завершения с ошибкой



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	R_TRIG_instance	R_TRIG		Экземпляр команды R_TRIG
	UP_Q	BOOL	ЛОЖЬ	Выход условия запуска
	FTPRemoveFile_instance	FTPRemoveFile		Экземпляр команды FTPRemoveFile

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	FTPRemoveDir_instance	FTPRemoveDir		Экземпляр команды FTPRemoveDir
	DoFTPTrigger	BOOL	ЛОЖЬ	Условие выполнения для команд FTPRemoveFile и FTPRemoveDir
	FTPAddr	_sFTP_CONNECT_SVR	(Adr := '', PortNo := 0, UserName := '', Password := '')	Параметры подключенного сервера FTP
	RemoveResult	ARRAY[0..0] OF _sFTP_FILE_RESULT	[(Name := '', TxError := False, RemoveError := False, Reserved := [4(16#0)])]	Результаты удаления файла
	Stage	UINT	0	Этап выполнения команды
	Trigger	BOOL	ЛОЖЬ	Условие выполнения

```
// Подготовка параметров подключенного сервера FTP.
IF P_First_RunMode THEN
   FTPAddr.Adr      := '192.168.250.2';// Адрес
   FTPAddr.PortNo   := UINT#21;        // Номер порта
   FTPAddr.UserName := 'FtpUser';      // Имя пользователя
   FTPAddr.Password := '12345678';     // Пароль
END_IF;

// Прием условия выполнения.
R_TRIG_instance(Trigger, UP_Q);
IF ( (UP_Q = TRUE) AND (FTPRemoveFile_instance.Busy = FALSE) AND
      (FTPRemoveDir_instance.Busy = FALSE) ) THEN
   DoFTPTrigger := TRUE;
   Stage := INT#1;
   FTPRemoveFile_instance( // Инициализация экземпляра.
      Execute          := FALSE,
      ConnectSvr       := FTPAddr,
      SvrDirName       := '/Backup/yyyy-mm-dd',
      FileName         := '*.*',
      RemoveFileResult := RemoveResult) ;
   FTPRemoveDir_instance( // Инициализация экземпляра.
      Execute       := FALSE,
      ConnectSvr    := FTPAddr,
      SvrDirName    := '/Backup',
      RemoveDirName := 'yyyy-mm-dd') ;
END_IF;

IF (DoFTPTrigger = TRUE) THEN
   CASE Stage OF
      1 : // Выполнение команды FTPRemoveFile.
         FTPRemoveFile_instance(
            Execute          := TRUE,                // Выполнение
            ConnectSvr       := FTPAddr,             // Подключенный FTP-сервер
```

```
            SvrDirName       := '/Backup/yyyy-mm-dd', //Имя каталога на FTP-сервере
            FileName         := '*.*',                // Имя файла
            RemoveFileResult := RemoveResult) ;       // Результаты удаления файла
         IF (FTPRemoveFile_instance.Done = TRUE) THEN
            Stage := INT#2;  // К следующему этапу
         ELSIF (FTPRemoveFile_instance.Error = TRUE) THEN
            Stage := INT#10; // Завершение с ошибкой
         END_IF;
      2 : // Выполнение команды FTPRemoveDir.
         FTPRemoveDir_instance(
            Execute       := TRUE,          // Выполнение
            ConnectSvr    := FTPAddr,       // Подключенный FTP-сервер
            SvrDirName    := '/Backup',     // Имя каталога на FTP-сервере
            RemoveDirName := 'yyyy-mm-dd') ;// Каталог для удаления
         IF (FTPRemoveDir_instance.Done = TRUE) THEN
            Stage:=INT#0;  // Нормальное завершение
         ELSIF (FTPRemoveDir_instance.Error = TRUE) THEN
            Stage:=INT#20; // Завершение с ошибкой
         END_IF;
      0 :  // Обработка после нормального завершения
         DoFTPTrigger:=FALSE;
         Trigger     :=FALSE;
      ELSE // Обработка после завершения с ошибкой
         DoFTPTrigger:=FALSE;
         Trigger     :=FALSE;
   END_CASE;
END_IF;
```

FTPRemoveDir

Команда FTPRemoveDir удаляет каталог с сервера FTP.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FTPRemoveDir	Удаление каталога с сервера FTP	FB	FTPRemoveDir_instance FTPRemoveDir Execute — Done ConnectSvr — Busy SvrDirName — CommandCanceled RemoveDirName — Error RetryCfg — ErrorID Cancel — ErrorIDEx	FTPRemoveDir_instance(Execute, ConnectSvr, SvrDirName, RemoveDirName, Cancel, RetryCfg, Done,Busy, CommandCanceled, Error, ErrorID, ErrorIDEx);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
ConnectSvr	Параметры подключенного сервера FTP	Вход	Параметры подключенного сервера FTP	---	---	*1
SvrDirName	Имя каталога сервера FTP		Имя каталога на сервере FTP, содержащего удаляемый каталог	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)*2		""*3
RemoveDirName	Удаляемый каталог		Удаляемый каталог	Макс. 256 байт (255 однобайтовых буквенно-цифровых символов + последний символ NULL)		*1
RetryCfg	Настройка повтора выполнения		Настройка повторных попыток выполнения команды	---		---
Cancel	Отмена		ИСТИНА: выполнение команды отменено. ЛОЖЬ: выполнение команды не отменено.	Зависит от типа данных.		ЛОЖЬ

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
CommandCanceled	Отмена завершена	Выход	ИСТИНА: отмена завершена. ЛОЖЬ: отмена не завершена.	---	---	---

*1. Если опустить какой-либо входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.
*2. В именах каталогов сервера FTP нельзя использовать следующие символы: * ? < > | “
*3. При входе на сервер FTP по умолчанию используется домашний каталог.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ConnectSvr	Подробные сведения о структуре _sFTP_CONNECT_SVR см. в разделе *Функция* на стр. 2-1370.																			
SvrDirName																				ОК
RemoveDirName																				ОК
RetryCfg	Подробные сведения о структуре _sFTP_RETRY_CFG см. в разделе *Настройка выполнения повторных попыток соединения с сервером FTP* на стр. 2-1343.																			
Cancel	ОК																			
CommandCanceled	ОК																			

Функция

Команда FTPRemoveDir удаляет указанный каталог (*RemoveDirName*) в указанном каталоге (*SvrDirName*) на подключенном сервере FTP (*ConnectSvr*).

Значение переменной *Done* команды меняется на ИСТИНА уже после того, как удаление указанного каталога завершено.
Если команде не удается удалить каталог, значение переменной *Error* меняется на ИСТИНА.

Для переменной *ConnectSvr* используется структурный тип данных _sFTP_CONNECT_SVR. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
ConnectSvr	Параметры подключенного сервера FTP	Параметры подключенного сервера FTP	_sFTP_CONNECT_SVR	---	---	---
Adr	Адрес	IP-адрес или имя хоста*1	STRING	1...200 байт*2	---	---
PortNo	Номер порта	Номер порта TCP сервера FTP для соединения	UINT	0...65535*3		
UserName	Имя пользователя	Имя пользователя на сервере FTP	STRING	Макс. 33 байт*4*5*6		
Password	Пароль	Пароль для входа на сервер FTP	STRING	Макс. 33 байт*4*5*6		

*1. Чтобы можно было указать имя хоста, должны быть отдельно настроены параметры DNS или хостов.
*2. Можно использовать следующие однобайтовые символы: "A...Z", "a...z", "0...9", "-" (дефис), "." (точка) и "_" (подчеркивание).
*3. Если указано значение 0, используется порт TCP с номером 21.
*4. Можно использовать следующие однобайтовые символы: "A...Z", "a...z", "0...9", "-" (дефис), "." (точка) и "_" (подчеркивание). Для модуля ЦПУ с версией модуля 1.16 или более поздней также можно использовать символы "\" (обратная косая черта) и "@".
*5. При подсчете количества байтов также следует учитывать нулевой символ (NULL).
*6. В случае модулей ЦПУ с версией модуля 1.08 следует указать текстовую строку, состоящую из одного или более символов. Если будет указана текстовая строка, содержащая только завершающий пустой символ (NULL), произойдет ошибка.

Настройка выполнения повторных попыток соединения с сервером FTP

Можно настроить выполнение повторных попыток соединения с сервером FTP.
Выполнение повторных попыток настраивается точно так же, как и для команды FTPGetFileList. См. *Настройка выполнения повторных попыток соединения с сервером FTP* на стр. 2-1325 для команды FTPGetFileList.

Отмена выполнения команды

Начавшееся выполнение команды FTPRemoveDir можно отменить.
Операция отмены производится точно так же, как и для команды FTPGetFileList. См. *Отмена выполнения команды* на стр. 2-1327 для команды FTPGetFileList.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_EIP_EtnOnlineSta*1	В сети	BOOL	Эта переменная показывает, когда могут использоваться функции связи встроенного порта EtherNet/IP. ИСТИНА: связь возможна. ЛОЖЬ: связь невозможна.
_EIP1_EtnOnlineSta*2 _EIP2_EtnOnlineSta*3			

*1. Используйте это имя переменной для модуля ЦПУ серии NJ.

*2. Используйте это имя переменной для порта 1 модуля ЦПУ серии NX.
Вместо _EIP1_EtnOnlineSta можно указать _EIP_EtnOnlineSta.
*3. Используйте это имя переменной для порта 2 модуля ЦПУ серии NX.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Эту команду можно использовать для встроенного порта EtherNet/IP Серия NJ/NX, модули ЦПУ.
- Выполнение этой команды будет продолжаться до завершения обработки, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении обработки значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- В некоторых случаях каталог на сервере FTP удаляется, даже если используется вход *Cancel* для отмены выполнения этой команды. Это зависит от того, в какой момент времени вход *Cancel* переводится в состояние ИСТИНА. Проверьте каталог на сервере FTP.
- Допускается одновременное выполнение не более 3 экземпляров следующих команд: FTPGetFileList, FTPGetFile, FTPPutFile, FTPRemoveFile и FTPRemoveDir.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение какого-либо входного параметра находится за пределами допустимого диапазона.
  b) На сервере FTP отсутствует каталог с указанным именем *SvrDirName*.
  c) ".." указано для уровня каталога в *SvrDirName* или *RemoveDirName*.
  d) Для *SvrDirName* или *RemoveDirName* указан неверный путь, например "//".
  e) На сервере FTP отсутствует каталог с указанным именем *RemoveDirName*.
  f) В каталоге с указанным именем *RemoveDirName* нет файлов или каталогов.
  g) Для каталога с указанным именем *RemoveDirName* установлен атрибут «Только чтение».
  h) Сервер FTP, указанный параметром *ConnectSvr*, отсутствует в сети или не работает.
  i) Было выполнено более 3 следующих команд одновременно: FTPGetFileList, FTPGetFile, FTPPutFile, FTPRemoveFile и FTPRemoveDir.
- Для этой команды дополнительный код ошибки *ErrorIDEx* содержит код ответа FTP, который был возвращен сервером FTP. В следующей таблице перечислены типичные значения дополнительного кода ошибки *ErrorIDEx*, описано значение ошибок и приведены возможные способы их устранения. Дополнительные сведения см. в документации по серверу FTP. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = WORD#16#2407.

Значение *ErrorIDEx*	Значение	Способ устранения
16#000001A9	Не удалось установить соединение для передачи данных.	В случае использования протокола FTP для обмена данными с сервером FTP через Интернет проследите, чтобы для параметра Open mode (Режим открытия) для FTP не было установлено значение Active (Активный).
16#000001AA	Соединение было закрыто. Передача данных была прервана.	Проверьте соединение с сервером FTP. Убедитесь, что сервер FTP работает.
16#000001C2	Не удалось выполнить запрошенную операцию с файлом. Использовать файл было невозможно, например, из-за того, что он уже был открыт.	Убедитесь, что файл не открыт для какого-нибудь другого приложения.

Значение *ErrorIDEx*	Значение	Способ устранения
16#00000212	Вход пользователя на сервер был невозможен.	Проверьте имя пользователя и пароль для входа на сервер FTP.
16#00000214	Для сохранения файлов требуется учетная запись.	Проверьте наличие у пользователя прав доступа к FTP.
16#00000226	Выполнить запрошенную операцию с файлом было невозможно, так как было невозможно использовать файл (например, не удалось получить доступ к файлу из-за того, что файл не был найден).	Убедитесь, что файл с указанным именем имеется в соответствующем каталоге на сервере FTP. Проверьте права доступа к указанному файлу.
16#00000229	Выполнение было невозможно, так как имя файла было неверным.	Проверьте права доступа к указанному каталогу.

Пример программы

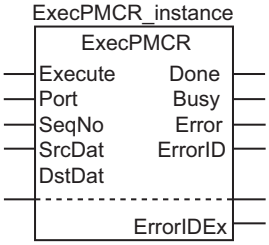
См. *Пример программы* на стр. 2-1364 для команды FTPRemoveFile.

Команды для обмена данными по последовательному интерфейсу

Команда	Имя	Стр.
ExecPMCR	Макрос протокола	стр. 2-1376
SerialSend	Передача через последовательный порт SCU	стр. 2-1391
SerialRcv и SerialRcvNoClear	Прием через последовательный порт SCU/Прием через последовательный порт SCU без очистки буфера приема	стр. 2-1403
SendCmd	Передача команды	стр. 2-1419
NX_SerialSend	Передача данных в беспротокольном режиме	стр. 2-1432
NX_SerialRcv	Прием данных в беспротокольном режиме	стр. 2-1445
NX_ModbusRtuCmd	Передача команды общего назначения по протоколу Modbus RTU	стр. 2-1460
NX_ModbusRtuRead	Передача команды чтения по протоколу Modbus RTU	стр. 2-1472
NX_ModbusRtuWrite	Передача команды записи по протоколу Modbus RTU	стр. 2-1485
NX_SerialSigCtl	Включение/выключение сигнала управления последовательного порта	стр. 2-1498
NX_SerialSigRead	Чтение сигнала управления последовательного порта	стр. 2-1506
NX_SerialStatusRead	Чтение состояния последовательного порта	стр. 2-1511
NX_SerialBufClear	Очистка буфера	стр. 2-1516
NX_SerialStartMon	Запуск мониторинга линии последовательного интерфейса	стр. 2-1527
NX_SerialStopMon	Остановка мониторинга линии последовательного интерфейса	стр. 2-1532

ExecPMCR

Команда ExecPMCR запрашивает выполнение коммуникационной последовательности (макроса протокола), зарегистрированной в модуле последовательного интерфейса.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ExecPMCR	Макрос протокола	FB		ExecPMCR_instance(Execute, Port, SeqNo, SrcDat, DstDat, Done, Busy, Error, ErrorID, ErrorIDEx);

Меры предосторожности для обеспечения надлежащей эксплуатации

Данную команду невозможно использовать с модулями ЦПУ серии NX.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Port	Порт назначения	Вход	Порт назначения	---	---	---
SeqNo	Номер коммуникационной последовательности		Номер коммуникационной последовательности	0...999	---	0
SrcDat[] (массив)	Массив передаваемых данных		Массив передаваемых данных	Зависит от типа данных.		*1
DstDat[] (массив)	Массив принимаемых данных	Вход-выход	Массив принимаемых данных	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Port	Подробные сведения о структуре _sPORT см. в разделе *Функция* на стр. 2-1377.																			
SeqNo							ОК													
SrcDat[] (массив)			ОК																	
DstDat[] (массив)			ОК																	

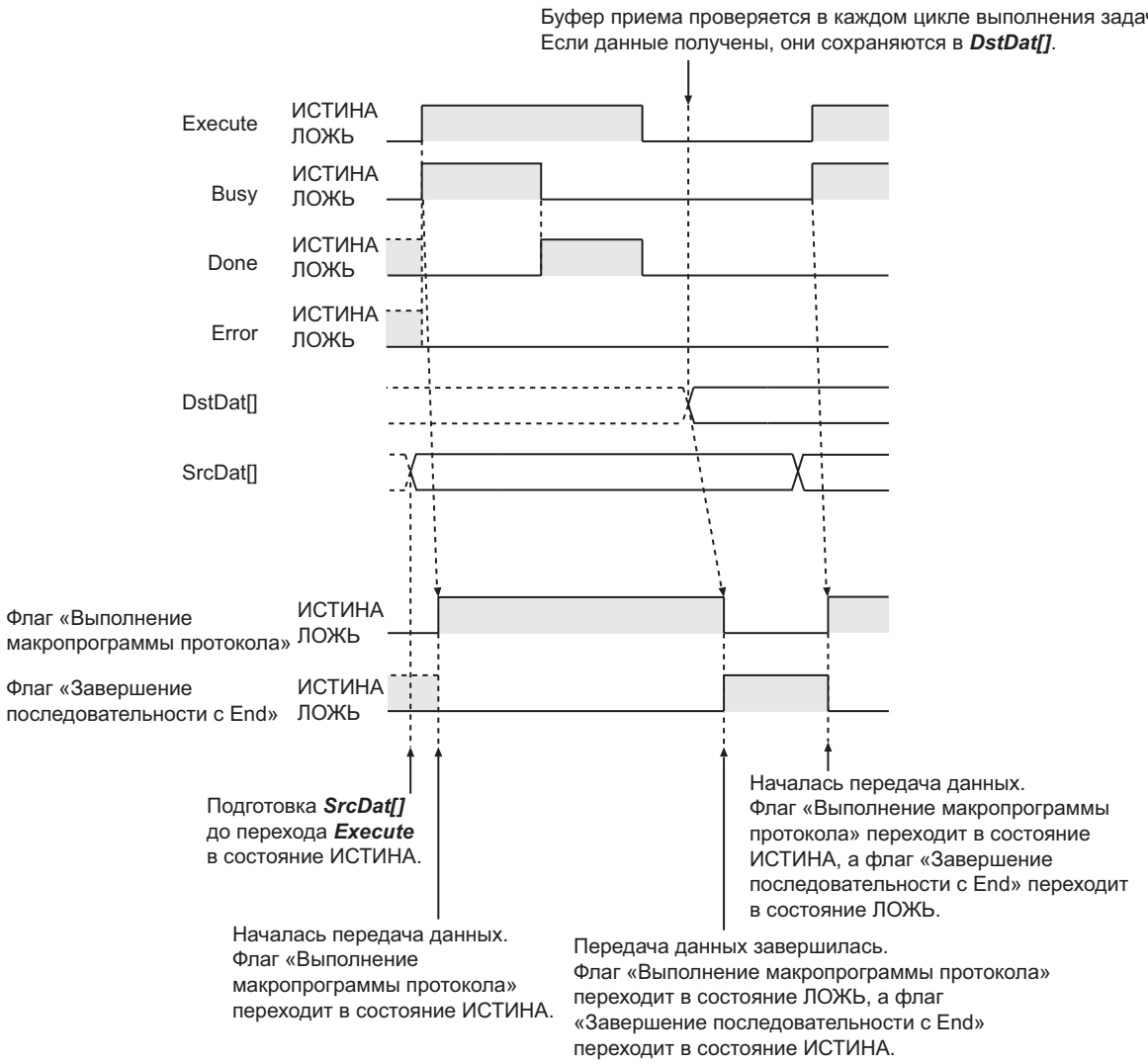
Функция

Команда ExecPMCR запрашивает выполнение коммуникационной последовательности с указанным номером (*SeqNo*) для указанного порта назначения (*Port*).
Если производится передача данных, передаются данные массива SrcDat[] (массив передаваемых данных), начиная с его 2-го элемента (SrcDat[1]). Количество элементов массива, данные которых нужно передать, указывается в элементе SrcDat[0].
Если прием данных выполнен успешно, принятые данные сохраняются в массив DstDat[] (массив принимаемых данных), начиная с его 2-го элемента (DstDat[1]). В элемент DstDat[0] записывается количество элементов с принятыми данными.
Если данные принять не удалось, то в элементах массива DstDat[], количество которых указано в DstDat[0], сохраняется содержимое, которое в них находилось до выполнения команды.

Для переменной *Port* (порт назначения) используется структурный тип данных _sPORT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Port	Порт назначения	Порт назначения	_sPORT	---	---	---
UnitNo	Номер модуля	Номер модуля для модуля последовательного интерфейса	_eUnitNo	_CBU_No00... _CBU_No15	---	_CBU_No00
PhysicPortNo	Номер последовательного порта	Номер последовательного порта на модуле последовательного интерфейса	USINT	1 или 2		1

На рисунке ниже представлена временная диаграмма, демонстрирующая работу команды. После перехода выхода *Done* в состояние ИСТИНА обмен данными продолжается и выполняется до конца.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Port_numUsingPort	Количество используемых портов	USINT	Количество портов, которые используются в настоящее время.
_Port_isAvailable	Флаг активации команд сетевой связи	BOOL	ИСТИНА: порт доступен. ЛОЖЬ: порт недоступен.
_CJB_SCU##P1ChgSta, _CJB_SCU##P2ChgSta*1	Флаг изменения параметров портов 1/2, модуль последовательного интерфейса ##	BOOL	ИСТИНА: в настоящее время изменяются параметры последовательного порта. ЛОЖЬ: в настоящее время параметры последовательного порта не изменяются.

*1. "##" — следует подставить номер модуля для модуля последовательного интерфейса.

Связанные переменные, частично определяемые пользователем

Имя	Значение	Тип данных	Описание
P#_PmrExecSta*1	Флаг выполнения макроса протокола	BOOL	ИСТИНА: в данный момент выполняется макрос протокола. ЛОЖЬ: в данный момент макрос протокола не выполняется.
P#_PmrSeqEndSta*1	Флаг «Завершение последовательности с End»	BOOL	ИСТИНА: последовательность была завершена с состоянием End. ЛОЖЬ: последовательность не была завершена с состоянием End.
P#_PmrSeqAbtSta*1	Флаг «Завершение последовательности с Abort»	BOOL	ИСТИНА: последовательность была завершена с состоянием Abort. ЛОЖЬ: последовательность не была завершена с состоянием Abort.

*1. "#" — следует подставить номер порта модуля последовательного интерфейса.

Дополнительная информация

Дополнительную информацию о макросах протоколов см. в документе *SYSMAC — CX-Protocol. Руководство по работе (Cat. No. W344)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Команда ExecPMCR запускает выполнение макроса протокола. Используйте системную переменную *P#PmrExecSta* (флаг выполнения макроса протокола) для проверки состояния выполнения макроса протокола.
- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- В атрибуте «AT» переменной DstDat[] должен быть указан адрес памяти, используемой для модулей серии CJ.
- Для непосредственного указания адреса и указания слова связи в элементы SrcDat[0] и DstDat[0] следует ввести значение 0. Если будет задана любая другая константа или переменная, произойдет ошибка и команда не будет выполнена.
- Если значение DstDat[0] = 0 или 1 и прием данных завершается сбоем, все элементы в массиве DstDat[] принимают значение 0.
- Команда выполняется только при наличии доступного порта. Поэтому системную переменную *_Port_isAvailable* (флаг активации команд сетевой связи) следует использовать для команды в качестве нормально разомкнутого условия выполнения.
- Команда не выполняется, пока переменная *Busy* = ИСТИНА. Поэтому переменную *Busy* следует использовать для команды в качестве нормально замкнутого условия выполнения.
- Когда начинается выполнение команды, частично определяемая пользователем переменная *P#_PmrExecSta* (флаг выполнения макроса протокола) принимает значение ИСТИНА. После того как коммуникационная последовательность будет выполнена и принятые данные будут сохранены в массив DstDat[], она снова примет значение ЛОЖЬ. До той поры эту команду невозможно выполнить для того же последовательного порта. Поэтому переменную

P#_PmrExecSta следует использовать для команды в качестве нормально замкнутого условия выполнения.

- Если команда используется в программе на языке ST, необходимо обеспечить, чтобы команда обрабатывалась в каждом цикле выполнения задачи в течение всего времени ее выполнения. В противном случае нормальная обработка может оказаться невозможной.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) При выполнении команды в качестве режима последовательной связи не установлен режим «Макрос протокола».
  b) Значение переменной *_Port_isAvailable* = ЛОЖЬ.
  c) Значение *SeqNo* вне допустимого диапазона.
  d) Коммуникационная последовательность с указанным номером *SeqNo* не зарегистрирована в модуле последовательного интерфейса.
  e) Значение *Port.UnitNo* или *Port.PhysicPortNo* вне допустимого диапазона.
  f) Модуль последовательного интерфейса серии CJ с указанным номером модуля отсутствует.
  g) Значение SrcDat[0] превосходит размер массива SrcDat[].
  h) Значение DstDat[0] превосходит размер массива DstDat[].
  i) Длина значения SrcDat[0] или DstDat[0] превышает 250 слов.
  j) Сбой связи.
  k) В атрибуте «AT» переменной DstDat[] не указан адрес памяти, используемой для модулей серии CJ.
- Для этой команды в качестве дополнительного кода ошибки в переменной *ErrorIDEx* указывается код ответа интерфейса связи.
  В следующей таблице приведены возможные значения кода ошибки и поясняется, что они означают. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = WORD#16#0800.

Значение	Ошибка	Способ устранения
16#00000001	Обслуживание связи было прервано.	• Проверьте состояние работы логических связей. • Проверьте емкость адресуемой области памяти, в которую передаются данные, на стороннем узле.
16#00000101	Локальный узел не является частью сети.	Сделайте локальный узел частью сети.
16#00000102	Превышено время ожидания маркера.	Задайте адрес локального узла так, чтобы он не превышал максимальный адрес узла.
16#00000103	Превышено максимальное количество повторных попыток.	Протестируйте связь между узлами. При обнаружении какой-либо ошибки проверьте программную и аппаратную инфраструктуру.
16#00000104	Превышено допустимое количество кадров передачи.	Проверьте состояние событий в сети и уменьшите число событий, приходящихся на каждый цикл выполнения задачи. Либо увеличьте допустимое количество кадров передачи.
16#00000105	IP-адрес локального узла вне допустимого диапазона.	Правильно настройте поворотные переключатели на модуле последовательного интерфейса.
16#00000106	IP-адрес локального узла также используется другим узлом в сети.	Измените дублирующийся адрес на одном из узлов.
16#00000201	Удаленный узел не является частью сети.	Сделайте удаленный узел частью сети.
16#00000202	Модуля с указанным адресом модуля не существует в месте назначения.	Правильно задайте адрес модуля для адреса сети назначения.

Значение	Ошибка	Способ устранения
16#00000203	Сторонний узел не является частью сети.	• Проверьте адрес модуля, который является сторонним узлом. • Укажите только один узел в качестве стороннего узла.
16#00000204	Удаленный узел занят.	Увеличьте количество повторных попыток или скорректируйте систему, чтобы коммуникационный трафик не концентрировался на удаленном узле.
16#00000205	Превышено время ожидания ответа.	Проверьте настройку параметров связи.
16#00000206	Имеется ошибка в канале передачи данных.	• Выполните операцию повторно. • Если эта ошибка возникает часто, проверьте наличие помех.
16#00000301	Произошла ошибка контроллера связи.	См. руководство по эксплуатации для соответствующего модуля и внесите необходимые исправления.
16#00000302	Ошибка модуля ЦПУ на удаленном узле.	См. документацию по модулю ЦПУ на удаленном узле и устраните ошибку.
16#00000303	В соответствующем контроллере произошла ошибка, и ответ не был возвращен.	Проверьте состояние связи в сети и перезапустите соответствующий контроллер. Если ошибка не исчезла, замените соответствующий контроллер.
16#00000304	Неверно задан номер модуля.	Правильно настройте поворотные переключатели на модуле последовательного интерфейса.
16#00000401	Переданная команда не поддерживается.	Правильно настройте массив команды.
16#00000402	Модель или версия модуля не поддерживаются.	Проверьте модель и версию модуля.
16#00000501	Неверно настроен удаленный адрес.	Задайте адрес назначения в таблице маршрутизации.
16#00000502	Таблицы маршрутизации не зарегистрированы.	Задайте исходный узел, узел назначение и промежуточные узлы в таблицах маршрутизации.
16#00000503	Имеется ошибка в таблицах маршрутизации.	Исправьте настройки в таблицах маршрутизации.
16#00000504	Слишком много промежуточных точек.	Измените конструкцию сети или исправьте таблицы маршрутизации таким образом, чтобы команды передавалась в пределах трех уровней сети.
16#00001001	Слишком длинная команда.	Правильно настройте массив команды.
16#00001002	Слишком короткая команда.	Правильно настройте массив команды.
16#00001003	Количество записываемых элементов, указанное в команде, не согласуется с объемом записываемых данных.	Задайте одинаковое количество записываемых элементов и записываемых данных.
16#00001004	Неверный формат команды.	Правильно настройте массив команды.
16#00001005	Имеется ошибка в заголовке.	Исправьте настройки в таблицах маршрутизации.
16#00001101	Тип области не существует.	См. информацию о переменных команды и кодах типов параметров и задайте подходящие коды.
16#00001102	Неверный размер для доступа к данным.	Правильно задайте размеры данных для доступа к переменным и параметрам.
16#00001103	Указан адрес вне допустимого диапазона.	Укажите адрес в пределах допустимого диапазона.
16#00001104	Превышен диапазон адресов.	• Укажите адрес в пределах допустимого диапазона. • Исправьте настройки в таблице логических связей.
16#00001106	Указан незарегистрированный номер коммуникационной последовательности.	Исправьте номер коммуникационной последовательности или добавьте последовательность с помощью CX-Protocol.

Значение	Ошибка	Способ устранения
16#00001109	Произошла ошибка взаимосвязи.	• Устраните несоответствие размеров друг другу в данных команды. • Исправьте настройки в таблице логических связей.
16#0000110A	Данные избыточны.	• Отмените текущий процесс или дождитесь его завершения, прежде чем выполнять команду. • Исправьте настройки в таблице логических связей.
16#0000110B	Слишком длинный ответ.	Правильно задайте количество элементов в массиве команды.
16#0000110C	Другая ошибка параметров.	Правильно настройте массив команды.
16#00002002	Данные защищены.	Снимите защиту и выполните команду еще раз.
16#00002003	Ни одна таблица не зарегистрирована.	Правильно настройте таблицу.
16#00002004	Нет данных, соответствующих искомым данным.	Правильно настройте данные для поиска.
16#00002005	Соответствующего номера программы не существует.	Задайте действительный номер программы.
16#00002006	Соответствующего файла не существует.	Правильно задайте имя файла, включая имена подкаталогов.
16#00002007	Произошла ошибка проверки.	• Проверьте содержимое памяти и запишите правильные данные. • Проверьте содержимое файла.
16#00002101	Доступ невозможен, так как область предназначена только для чтения.	Снимите защиту от записи и выполните команду еще раз.
16#00002102	Данные защищены или запись в таблицу логических связей невозможна.	• Снимите защиту от записи и выполните команду еще раз. • Задайте системные параметры в таблице логических связей.
16#00002103	Регистрация невозможна.	• Создайте файл после удаления ненужных файлов или подготовьте новую файловую память. • Закройте открытые файлы и выполните команду еще раз.
16#00002105	Соответствующего номера программы не существует.	Задайте действительный номер программы.
16#00002106	Соответствующего файла не существует.	Правильно задайте имя файла, включая имена подкаталогов.
16#00002107	Файл с таким именем уже существует.	Измените имя файла для записи и выполните команду еще раз.
16#00002108	Изменение недопустимо, так как оно вызывает ошибку.	Скорректируйте настройки.
16#00002201	Операция невозможна, так как макрос протокола уже выполняется.	Используйте флаг выполнения макроса протокола в качестве нормально замкнутого входа программы.
16#00002202	Неверный режим работы.	Проверьте режим работы.
16#00002203	Неверный режим работы для команды (режим «Программирование»).	Проверьте режим работы контроллера.
16#00002204	Неверный режим работы для команды (режим «Отладка»).	Проверьте режим работы контроллера.
16#00002205	Неверный режим работы для команды (режим «Мониторинг»).	Проверьте режим работы контроллера.
16#00002206	Неверный режим работы для команды (режим «Выполнение»).	Проверьте режим работы контроллера.
16#00002207	Указанный узел не является опрашивающим узлом.	Проверьте, какой из узлов сети является опрашивающим узлом.

Значение	Ошибка	Способ устранения
16#00002208	Неверный режим работы для команды.	Проверьте состояние активации шага.
16#00002211	Модуль занят.	Увеличьте количество повторных попыток или скорректируйте систему, чтобы коммуникационный трафик не концентрировался на соответствующем модуле.
16#00002301	Файлового устройства не существует.	Вставьте носитель. Или отформатируйте память EM.
16#00002302	Файловой памяти нет.	Проверьте устройство файловой памяти.
16#00002303	Встроенных часов нет.	Проверьте технические характеристики модели.
16#00002401	Ошибка контрольной суммы в данных макроса протокола или передача данных еще не завершена.	Передайте данные макроса протокола из CX-Protocol еще раз.
16#00002502	Имеется ошибка в памяти.	Передайте в память правильные данные.
16#00002503	Зарегистрированная конфигурация модулей ввода-вывода не соответствует фактической конфигурации модулей.	Проверьте конфигурацию модулей ввода-вывода.
16#00002504	Слишком много локальных или удаленных точек ввода-вывода.	Правильно задайте количество локальных и удаленных точек ввода-вывода.
16#00002505	Произошла ошибка при обмене данными между модулем ЦПУ и модулем шины ЦПУ.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#00002506	Один и тот же номер стойки, номер модуля или адрес ввода/вывода задан более одного раза.	Исправьте настройки, чтобы каждый номер был уникальным.
16#00002507	Произошла ошибка при обмене данными между модулем ЦПУ и модулем ввода-вывода.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#00002509	Ошибка передачи данных в SYSMAC BUS/2.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#0000250A	Произошла ошибка при передаче данных модуля шины ЦПУ.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#0000250D	Один и тот же канал используется более одного раза.	Правильно настройте каналы ввода-вывода.
16#0000250F	Имеется ошибка в памяти.	• В случае внутренней памяти: запишите правильные данные и выполните команду еще раз. • В случае карты памяти или файловой памяти EM: выполните команду форматирования памяти расширения. • Если указанные выше меры не помогают устранить ошибку, замените память.
16#00002510	Неверно настроена конечная станция.	Правильно задайте конечную станцию.
16#00002601	Защита уже снята.	Снимать защиту не требуется.
16#00002602	Введен неверный пароль.	Укажите правильный пароль.
16#00002604	Данные защищены.	• Снимите защиту от записи и выполните команду еще раз. • Дождитесь завершения службы, которая выполняется в данный момент, или остановите службу и снова выполните команду.

Значение	Ошибка	Способ устранения
16#00002605	Служба занята.	Дождитесь завершения службы, которая выполняется в данный момент, или остановите службу и снова выполните команду.
16#00002606	Служба остановлена.	При необходимости выполните соответствующую службу.
16#00002607	У вас нет права на выполнение.	• Выполните операцию с узла, с которого был произведен доступ к каналу данных. • Если ошибка все еще присутствует после перезапуска, замените контроллер.
16#00002608	Окружение не настроено.	Выполните необходимую настройку.
16#00002609	Необходимые параметры не заданы.	Задайте необходимые параметры.
16#0000260A	Указанный номер уже определен.	Укажите номер действия или номер перехода, который еще не зарегистрирован, и выполните команду еще раз.
16#0000260B	Невозможно сбросить ошибку.	Устраните причину ошибки, а затем выполните команду сброса ошибки.
16#00003001	У вас нет прав доступа.	Подождите, пока будет разрешен доступ, а затем снова выполните команду.
16#00004001	Служба была прервана.	Устраните причину, по которой была прервана служба, и выполните команду еще раз.

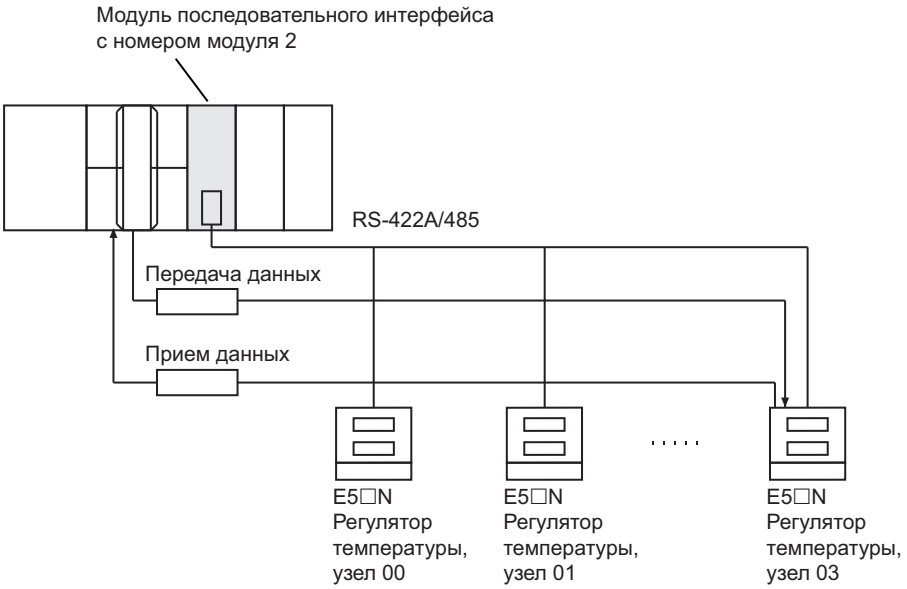
Примечание Помимо кодов, приведенных в таблице выше, о наличии ошибки могут свидетельствовать биты 6, 7 и 15 в коде завершения, которые принимают значение ИСТИНА в случае ошибки. Если значение бита 6 или 7 = ИСТИНА, значит имеется ошибка в модуле ЦПУ в месте назначения. Если значение бита 15 = ИСТИНА, значит произошла ошибка при промежуточной передаче данных по сети.

Пример программы

В данном примере модуль последовательного интерфейса серии CJ используется для обмена данными с регулятором температуры OMRON. Из регулятора температуры считывается текущее значение с помощью макроса протокола. Используется коммуникационная последовательность для CompoWay/F Master под номером 610 (Чтение области переменных).
Из контроллера в регулятор передается содержимое массива передаваемых данных SendData[].
Данные, принятые от регулятора температуры, сохраняются в массив принятых данных RecvData[].

В таблице ниже указаны используемые модули и параметры связи.

Параметр	Описание
Используемый модуль	Модуль последовательного интерфейса
Номер модуля	2
Номер порта	1 (RS-422/485)
Номер коммуникационной последовательности	610 (Чтение области переменных)
Номер удаленного узла	3
Данные для чтения	Текущее значение



Ниже показано распределение данных связи для последовательности 610 (Чтение области переменных).

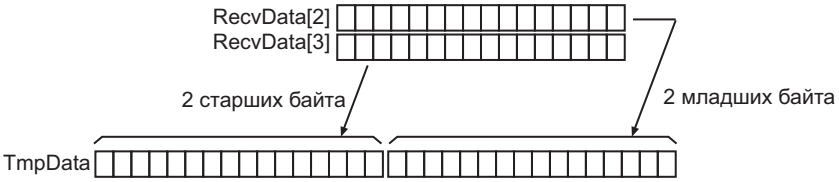
Передаваемые данные:
массив типа WORD

SendData[0]	Кол-во передав. слов данных
SendData[1]	Не использ. / Номер узла
SendData[2]	Тип переменной
SendData[3]	Начальный адрес чтения
SendData[4]	Количество элементов

Принимаемые данные:
массив типа WORD

RecvData[0]	Кол-во приним. слов данных
RecvData[1]	Код ответа
RecvData[2]	Принятые данные
RecvData[3]	

В случае успешного приема данных переменной *TmpData* присваиваются два младших байта (RecvData[2]) и два старших байта (RecvData[3]) текущего значения, прочитанного из регулятора температуры.



Передаваемые данные SendData[] и принятые данные RecvData[]

Ниже приведено содержимое массивов SendData[] (передаваемые данные) и RecvData[] (принятые данные).

● Передаваемые данные: массив типа WORD

Переменная	Параметр	Содержание	Значение
SendData[0]	Количество передаваемых слов данных	Передается пять слов данных: SendData[0]...SendData[4].	WORD#16#0005
SendData[1]	Номер узла	Используется узел 3.	WORD#16#0003

Переменная	Параметр	Содержание	Значение
SendData[2]	Тип переменной + старший байт начального адреса чтения	Тип переменной для чтения текущего значения: BYTE#16#C0; начальный адрес чтения: WORD#16#00.	WORD#16#C000
SendData[3]	Младший байт начального адреса чтения + BYTE#16#00 (фиксированное значение)		WORD#16#0000
SendData[4]	Количество элементов	Считывается один элемент.	WORD#16#0001

● Принятые данные: массив типа WORD

Переменная	Параметр	Содержание	Значение
RecvData[0]	Количество принятых слов данных	Принимается четыре слова данных: RecvData[0]...RecvData[3].	WORD#16#0004
RecvData[1]	Код ответа	При нормальном завершении возвращается WORD#16#0000.	
RecvData[2]	Принимаемые данные	Возвращаются два младших байта текущего значения регулятора температуры.	
RecvData[3]		Возвращаются два старших байта текущего значения регулятора температуры.	

Определения глобальных переменных

● Глобальные переменные

Имя	Тип данных	Параметр «AT»*1	Комментарий
SCU_P1_PmrSeqEndSta	BOOL	IOBus://rack#0/slot#0/P1_PmrSta/P1_PmrSeqEndSta	Флаг «Завершение последовательности с End»
SCU_P1_PmrSeqAbtSta	BOOL	IOBus://rack#0/slot#0/P1_PmrSta/P1_PmrSeqAbtSta	Флаг «Завершение последовательности с Abort»
SCU_P1_PmrExecSta	BOOL	IOBus://rack#0/slot#0/P1_PmrSta/P1_PmrExecSta	Флаг выполнения макроса протокола

*1. Параметр «AT» для случая, когда модуль последовательного интерфейса установлен в слот с номером 0 стойки с номером 0.

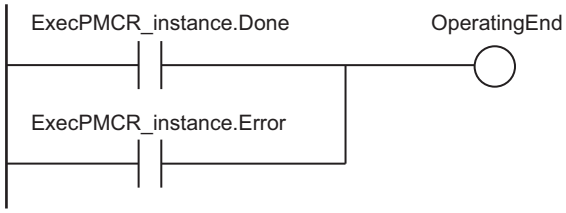
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	AT	Сохранение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ		☐	Обработка завершена
	Trigger	BOOL	ЛОЖЬ		☐	Условие выполнения
	Operating	BOOL	ЛОЖЬ		☐	Обработка

Внутренние переменные	Переменная	Тип данных	Начальное значение	AT	Сохранение	Комментарий
	InPort	_sPORT	(UnitNo:=_CBU_No00, PhysicPortNo:=0)		☐	Параметры порта
	SendData	ARRAY[0..4] OF WORD	[5(16#0)]		☐	Передаваемые данные
	RecvData	ARRAY[0..3] OF WORD	[4(16#0)]	%D200	☑	Принимаемые данные
	TmpData	DINT	0		☐	Текущее значение
	RS_instance	RS			☐	
	ExecPMCR_instance	ExecPMCR			☐	

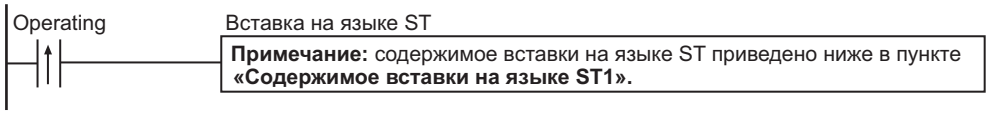
Внешние переменные	Переменная	Тип данных	Комментарий
	SCU_P1_PmrSeqEndSta	BOOL	Флаг «Завершение последовательности с End»
	SCU_P1_PmrSeqAbtSta	BOOL	Флаг «Завершение последовательности с Abort»
	SCU_P1_PmrExecSta	BOOL	Флаг выполнения макроса протокола
	_Port_isAvailable	BOOL	Флаг активации команд сетевой связи

Проверка завершения выполнения команды ExecPMCR.

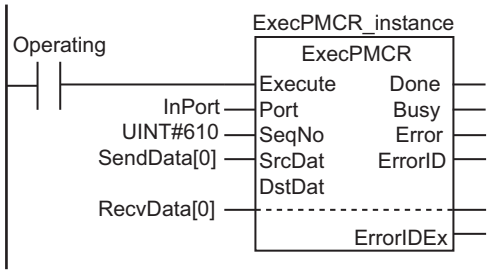


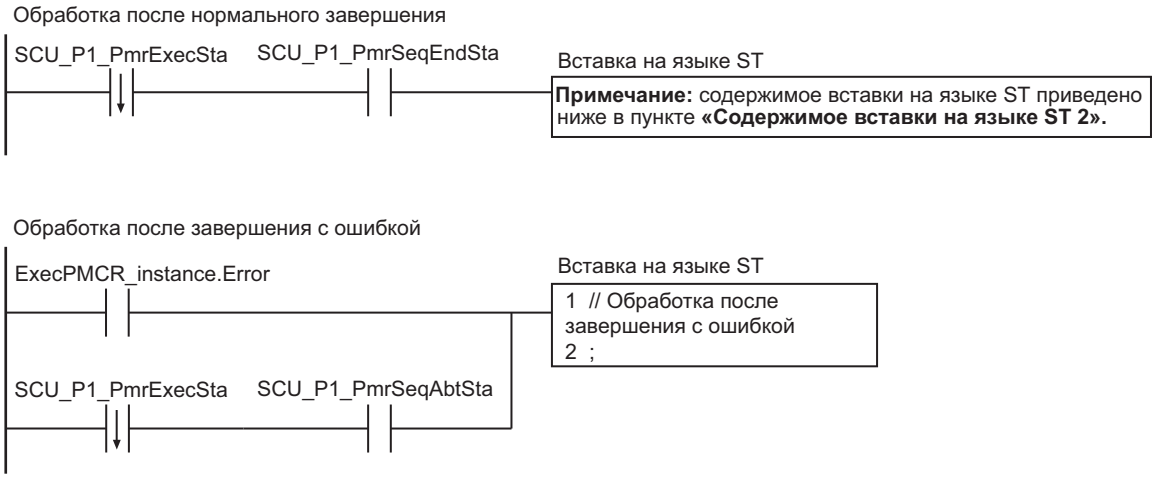
Прием условия выполнения.

Настройка параметров связи.



Выполнение команды ExecPMCR.





● Содержимое вставки на языке ST 1

```
InPort.UnitNo          :=_CBU_No02;     // Модуль послед. интерф. с номером модуля 2
InPort.PhysicPortNo    :=USINT#1;       // Номер порта 1
SendData[0]            :=WORD#16#0005;
SendData[1]            :=WORD#16#0003;
SendData[2]            :=WORD#16#C000;
SendData[3]            :=WORD#16#0000;
SendData[4]            :=WORD#16#0001;
RecvData[0]            :=WORD#16#0004;
```

● Содержимое вставки на языке ST 2

```
// Обработка после нормального завершения
TmpData:=DWORD_TO_DINT(SHL(WORD_TO_DWORD(
         RecvData[3]), 16) OR WORD_TO_DWORD(RecvData[2]) );
```

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	AT	Сохранение	Комментарий
	State	INT	0		☐	Текущее состояние
	Trigger	BOOL	ЛОЖЬ		☐	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ		☐	Значение Trigger в предыдущем цикле выполнения задачи
	InPort	_sPORT	(UnitNo:=_CBU_No00, PhysicPortNo:=0)		☐	Параметры порта
	SendData	ARRAY[0..4] OF WORD	[5(16#0)]		☐	Передаваемые данные
	RecvData	ARRAY[0..3] OF WORD	[4(16#0)]	%D200	☑	Принимаемые данные

Внутренние переменные	Переменная	Тип данных	Начальное значение	AT	Сохранение	Комментарий
	End_ExecPMCR	BOOL	ЛОЖЬ		☐	Завершение выполнения команды ExecPMCR
	TmpData	DINT	0		☐	Текущее значение
	RS_instance	RS			☐	
	ExecPMCR_instance	ExecPMCR			☐	
	F_TRIG_instance	F_TRIG			☐	

Внешние переменные	Переменная	Тип данных	Комментарий
	SCU_P1_PmrSeqEndSta	BOOL	Флаг «Завершение последовательности с End»
	SCU_P1_PmrSeqAbtSta	BOOL	Флаг «Завершение последовательности с Abort»
	SCU_P1_PmrExecSta	BOOL	Флаг выполнения макроса протокола
	_Port_isAvailable	BOOL	Флаг активации команд сетевой связи

```
// Прием условия выполнения.
IF (State=INT#0) THEN
    IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Port_isAvailable=TRUE) AND (S
CU_P1_PmrExecSta<>TRUE)
         AND (ExecPMCR_instance.Busy<>TRUE) ) THEN
       State:=INT#1;
    END_IF;
END_IF;
LastTrigger:=Trigger;

// Настройка параметров связи и инициализация команды ExecPMCR.
IF (State=INT#1) THEN
    InPort.UnitNo          :=_CBU_No02;    // Модуль послед. интерф. с номером моду
ля 2
    InPort.PhysicPortNo    :=USINT#1;      // Номер порта 1
    SendData[0]            :=WORD#16#0005;
    SendData[1]            :=WORD#16#0003;
    SendData[2]            :=WORD#16#C000;
    SendData[3]            :=WORD#16#0000;
    SendData[4]            :=WORD#16#0001;
    RecvData[0]            :=WORD#16#0004;
    ExecPMCR_instance(
        Execute :=FALSE,         // Инициализация команды ExecPMCR.
        SrcDat  :=SendData[0],   // Формальная запись
        DstDat  :=RecvData[0]);
     State:=INT#2;
END_IF;
// Выполнение команды ExecPMCR.
IF (State=INT#2) THEN
```

```
    ExecPMCR_instance(
    Execute :=TRUE,
    Port    :=InPort,
    SeqNo   :=UINT#610,
    SrcDat  :=SendData[0],
    DstDat  :=RecvData[0]);

    F_TRIG_instance(SCU_P1_PmrExecSta, End_ExecPMCR);

    IF (End_ExecPMCR=TRUE) THEN
        End_ExecPMCR:=FALSE;
        State:=INT#3;
    END_IF;

    IF (ExecPMCR_instance.Error=TRUE) THEN
        State:=INT#5;
    END_IF;
END_IF;

// Подтверждение завершения выполнения команды ExecPMCR.
IF (State=INT#3) THEN
    IF (SCU_P1_PmrSeqEndSta=TRUE) THEN
        State:=INT#4;
    END_IF;
    IF (SCU_P1_PmrSeqAbtSta=TRUE) THEN
        State:=INT#5;
    END_IF;
END_IF;

IF (State=INT#4) THEN
    // Обработка после нормального завершения.
    TmpData:=DWORD_TO_DINT(SHL(WORD_TO_DWORD(RecvData[3]), 16)
             OR WORD_TO_DWORD(RecvData[2]));
    State:=INT#0;
END_IF;

IF (State=INT#5) THEN
    // Обработка после завершения с ошибкой
    State:=INT#0;
END_IF;
```

SerialSend

Команда SerialSend выполняет передачу данных в беспротокольном режиме через последовательный порт модуля последовательного интерфейса.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SerialSend	Передача через последовательный порт SCU	FB	SerialSend_instance SerialSend Execute Done Port Busy SrcDat Error SendSize ErrorID ErrorIDEx	SerialSend_instance(Execute, Port, SrcDat, SendSize, Done, Busy, Error, ErrorID, ErrorIDEx);

Меры предосторожности для обеспечения надлежащей эксплуатации

Данную команду невозможно использовать с модулями ЦПУ серии NX.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Port	Порт назначения	Вход	Порт назначения	---	---	---
SrcDat[] (массив)	Массив передаваемых данных		Массив передаваемых данных	Зависит от типа данных.		*1
SendSize	Объем передаваемых данных		Объем данных массива SrcDat[], которые нужно передать	0...256	Байты	1

*1. Если опустить входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Port	Подробные сведения о структуре _sPORT см. в разделе *Функция* на стр. 2-1391.																			
SrcDat[] (массив)		ОК																		
SendSize							ОК													

Функция

Команда SerialSend выполняет передачу данных в беспротокольном режиме через порт модуля последовательного интерфейса, указанный параметром *Port*.

Данные, которые нужно передать, содержатся в массиве SrcDat[] (массив передаваемых данных). Объем передаваемых данных указывается с помощью параметра *SendSize* (объем передаваемых данных).

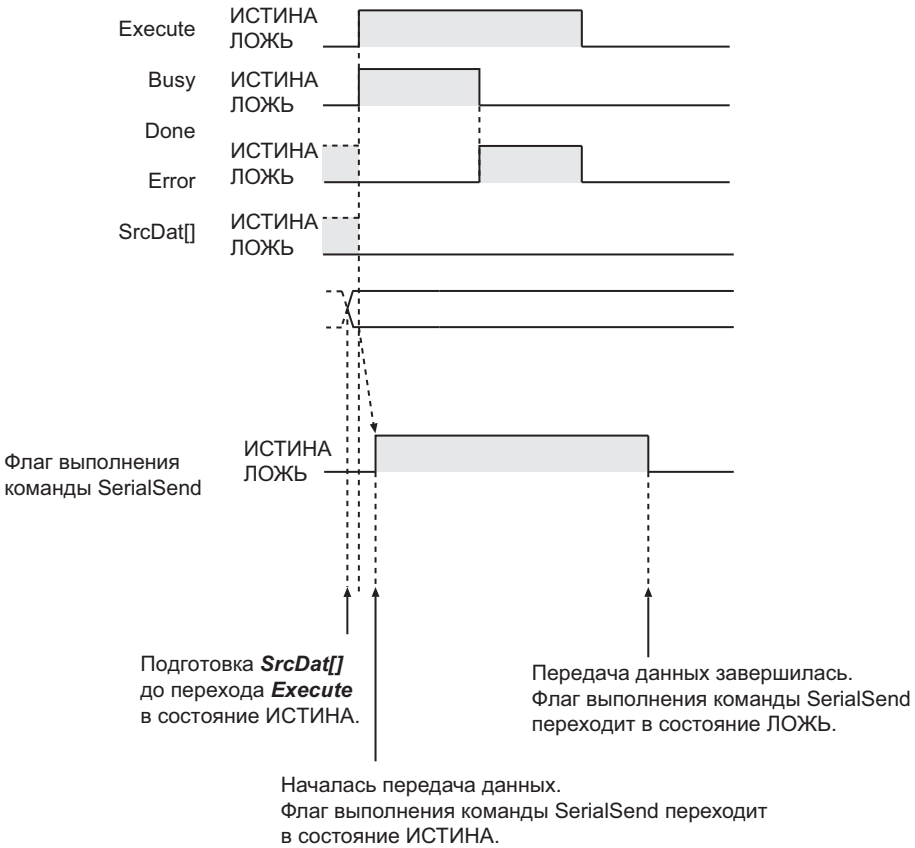
Чтобы присоединить к передаваемым данным код начала данных и код конца данных, эти коды следует задать в словах области DM, назначенных специальному модулю.

В случае добавления кодов начала и конца данных максимальное количество передаваемых байтов составляет 259 (1 байт кода начала данных, 2 байта кода конца данных (для указания кода CR+LF) и 256 байтов передаваемых данных).

Для переменной *Port* (порт назначения) используется структурный тип данных _sPORT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Port	Порт назначения	Порт назначения	_sPORT	---	---	---
UnitNo	Номер модуля	Номер модуля для модуля последовательного интерфейса	_eUnitNo	_CBU_No00..._CBU_No15	---	_CBU_No00
PhysicPortNo	Номер последовательного порта	Номер последовательного порта на модуле последовательного интерфейса	USINT	1 или 2		1

На рисунке ниже представлена временная диаграмма, демонстрирующая работу команды. После перехода выхода *Done* в состояние ИСТИНА обмен данными продолжается и выполняется до конца.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Port_numUsingPort	Количество используемых портов	USINT	Количество портов, которые используются в настоящее время.
_Port_isAvailable	Флаг активации команд сетевой связи	BOOL	ИСТИНА: порт доступен. ЛОЖЬ: порт недоступен.

Связанные переменные, частично определяемые пользователем

Имя	Значение	Тип данных	Описание
P#_NopSerialSendExecSta*1	Флаг выполнения команды SerialSend	BOOL	ИСТИНА: в данный момент выполняется команда SerialSend. ЛОЖЬ: команда SerialSend не выполняется в данный момент.
P#_NopStartCodeYNCfg*1	Активация кода начала данных в беспротокольном режиме	BOOL	ИСТИНА: использовать код начала ЛОЖЬ: не использовать код начала
P#_NopEndCodeYNCfg*1	Активация кода конца данных в беспротокольном режиме	BOOL	ИСТИНА: использовать код конца ЛОЖЬ: не использовать код конца
P#_NopCRLFCfg*1	Указание кода CR LF в беспротокольном режиме	BOOL	ИСТИНА: использовать CR+LF ЛОЖЬ: не использовать CR+LF
P#_NopStartCodeCfg*1	Код начала данных в беспротокольном режиме	USINT	16#00...16#FF
P#_NopEndCodeCfg*1	Код конца данных в беспротокольном режиме	USINT	16#00...16#FF

*1. "#" — следует подставить номер порта модуля последовательного интерфейса.

Дополнительная информация

Дополнительные сведения о передаче данных в беспротокольном режиме см. в следующем руководстве:

- *Серия CJ, модули последовательного интерфейса — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W494)*

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эту команду можно использовать только для последовательного порта модуля последовательного интерфейса, который был переведен в режим «Без протокола».
- Если значение *SendSize* равно 0, ничего не передается. При выполнении команды значение переменной *Done* меняется на ИСТИНА.
- При добавлении кодов начала и конца данных их не следует учитывать в значении *SendSize*.

- Команда выполняется только при наличии доступного порта. Поэтому системную переменную *_Port_isAvailable* (флаг активации команд сетевой связи) следует использовать для команды в качестве нормально разомкнутого условия выполнения.
- Команда не выполняется, пока переменная *Busy* = ИСТИНА. Поэтому переменную *Busy* следует использовать для команды в качестве нормально замкнутого условия выполнения.
- Эту команду невозможно выполнить, когда флаг выполнения команды SerialSend (частично определяемая пользователем переменная *P#NopSerialSendExecSta*) = ИСТИНА. Переменную *P#NopSerialSendExecSta* следует использовать для команды в качестве нормально замкнутого условия выполнения.
- Если команда используется в программе на языке ST, необходимо обеспечить, чтобы команда обрабатывалась в каждом цикле выполнения задачи в течение всего времени ее выполнения. В противном случае нормальная обработка может оказаться невозможной.
- В указанных ниже случаях происходит ошибка. Значение *Error* поменяется на ИСТИНА.
  a) При выполнении команды в качестве режима последовательной связи не установлен режим «Без протокола».
  b) Значение переменной *_Port_isAvailable* = ЛОЖЬ.
  c) Значение *Port.UnitNo* или *Port.PhysicPortNo* вне допустимого диапазона.
  d) Модуль последовательного интерфейса серии CJ с указанным номером модуля отсутствует.
  e) Значение *SendSize* вне допустимого диапазона.
  f) Значение *SendSize* превосходит размер массива SrcDat[].
  g) Сбой связи.
  h) Команда выполнена во время перезапуска модуля.
- Для этой команды в качестве дополнительного кода ошибки в переменной *ErrorIDEx* указывается код ответа интерфейса связи.
  В следующей таблице приведены возможные значения кода ошибки и поясняется, что они означают. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = WORD#16#0800.

Значение	Ошибка	Способ устранения
16#00000001	Обслуживание связи было прервано.	• Проверьте состояние работы логических связей. • Проверьте емкость адресуемой области памяти, в которую передаются данные, на стороннем узле.
16#00000101	Локальный узел не является частью сети.	Сделайте локальный узел частью сети.
16#00000102	Превышено время ожидания маркера.	Задайте адрес локального узла так, чтобы он не превышал максимальный адрес узла.
16#00000103	Превышено максимальное количество повторных попыток.	Протестируйте связь между узлами. При обнаружении какой-либо ошибки проверьте программную и аппаратную инфраструктуру.
16#00000104	Превышено допустимое количество кадров передачи.	Проверьте состояние событий в сети и уменьшите число событий, приходящихся на каждый цикл выполнения задачи. Либо увеличьте допустимое количество кадров передачи.
16#00000105	IP-адрес локального узла вне допустимого диапазона.	Правильно настройте поворотные переключатели на модуле последовательного интерфейса.
16#00000106	IP-адрес локального узла также используется другим узлом в сети.	Измените дублирующийся адрес на одном из узлов.
16#00000201	Удаленный узел не является частью сети.	Сделайте удаленный узел частью сети.
16#00000202	Модуля с указанным адресом модуля не существует в месте назначения.	Правильно задайте адрес модуля для адреса сети назначения.

Значение	Ошибка	Способ устранения
16#00000203	Сторонний узел не является частью сети.	• Проверьте адрес модуля, который является сторонним узлом. • Укажите только один узел в качестве стороннего узла.
16#00000204	Удаленный узел занят.	Увеличьте количество повторных попыток или скорректируйте систему, чтобы коммуникационный трафик не концентрировался на удаленном узле.
16#00000205	Превышено время ожидания ответа.	Проверьте настройку параметров связи.
16#00000206	Имеется ошибка в канале передачи данных.	• Выполните операцию повторно. • Если эта ошибка возникает часто, проверьте наличие помех.
16#00000301	Произошла ошибка контроллера связи.	См. руководство по эксплуатации для соответствующего модуля и внесите необходимые исправления.
16#00000302	Ошибка модуля ЦПУ на удаленном узле.	См. документацию по модулю ЦПУ на удаленном узле и устраните ошибку.
16#00000303	В соответствующем контроллере произошла ошибка, и ответ не был возвращен.	Проверьте состояние связи в сети и перезапустите соответствующий контроллер. Если ошибка не исчезла, замените соответствующий контроллер.
16#00000304	Неверно задан номер модуля.	Правильно настройте поворотные переключатели на модуле последовательного интерфейса.
16#00000401	Переданная команда не поддерживается.	Правильно настройте массив команды.
16#00000402	Модель или версия модуля не поддерживаются.	Проверьте модель и версию модуля.
16#00000501	Неверно настроен удаленный адрес.	Задайте адрес назначения в таблице маршрутизации.
16#00000502	Таблицы маршрутизации не зарегистрированы.	Задайте исходный узел, узел назначение и промежуточные узлы в таблицах маршрутизации.
16#00000503	Имеется ошибка в таблицах маршрутизации.	Исправьте настройки в таблицах маршрутизации.
16#00000504	Слишком много промежуточных точек.	Измените конструкцию сети или исправьте таблицы маршрутизации таким образом, чтобы команды передавалась в пределах трех уровней сети.
16#00001001	Слишком длинная команда.	Правильно настройте массив команды.
16#00001002	Слишком короткая команда.	Правильно настройте массив команды.
16#00001003	Количество записываемых элементов, указанное в команде, не согласуется с объемом записываемых данных.	Задайте одинаковое количество записываемых элементов и записываемых данных.
16#00001004	Неверный формат команды.	Правильно настройте массив команды.
16#00001005	Имеется ошибка в заголовке.	Исправьте настройки в таблицах маршрутизации.
16#00001101	Тип области не существует.	См. информацию о переменных команды и кодах типов параметров и задайте подходящие коды.
16#00001102	Неверный размер для доступа к данным.	Правильно задайте размеры данных для доступа к переменным и параметрам.
16#00001103	Указан адрес вне допустимого диапазона.	Укажите адрес в пределах допустимого диапазона.
16#00001104	Превышен диапазон адресов.	• Укажите адрес в пределах допустимого диапазона. • Исправьте настройки в таблице логических связей.
16#00001106	Указан незарегистрированный номер коммуникационной последовательности.	Исправьте номер коммуникационной последовательности или добавьте последовательность с помощью CX-Protocol.

Значение	Ошибка	Способ устранения
16#00001109	Произошла ошибка взаимосвязи.	• Устраните несоответствие размеров друг другу в данных команды. • Исправьте настройки в таблице логических связей.
16#0000110A	Данные избыточны.	• Отмените текущий процесс или дождитесь его завершения, прежде чем выполнять команду. • Исправьте настройки в таблице логических связей.
16#0000110B	Слишком длинный ответ.	Правильно задайте количество элементов в массиве команды.
16#0000110C	Другая ошибка параметров.	Правильно настройте массив команды.
16#00002002	Данные защищены.	Снимите защиту и выполните команду еще раз.
16#00002003	Ни одна таблица не зарегистрирована.	Правильно настройте таблицу.
16#00002004	Нет данных, соответствующих искомым данным.	Правильно настройте данные для поиска.
16#00002005	Соответствующего номера программы не существует.	Задайте действительный номер программы.
16#00002006	Соответствующего файла не существует.	Правильно задайте имя файла, включая имена подкаталогов.
16#00002007	Произошла ошибка проверки.	• Проверьте содержимое памяти и запишите правильные данные. • Проверьте содержимое файла.
16#00002101	Доступ невозможен, так как область предназначена только для чтения.	Снимите защиту от записи и выполните команду еще раз.
16#00002102	Данные защищены или запись в таблицу логических связей невозможна.	• Снимите защиту от записи и выполните команду еще раз. • Задайте системные параметры в таблице логических связей.
16#00002103	Регистрация невозможна.	• Создайте файл после удаления ненужных файлов или подготовьте новую файловую память. • Закройте открытые файлы и выполните команду еще раз.
16#00002105	Соответствующего номера программы не существует.	Задайте действительный номер программы.
16#00002106	Соответствующего файла не существует.	Правильно задайте имя файла, включая имена подкаталогов.
16#00002107	Файл с таким именем уже существует.	Измените имя файла для записи и выполните команду еще раз.
16#00002108	Изменение недопустимо, так как оно вызывает ошибку.	Скорректируйте настройки.
16#00002201	Операция невозможна, так как макрос протокола уже выполняется.	Используйте флаг выполнения макроса протокола в качестве нормально замкнутого входа программы.
16#00002202	Неверный режим работы.	Проверьте режим работы.
16#00002203	Неверный режим работы для команды (режим «Программирование»).	Проверьте режим работы контроллера.
16#00002204	Неверный режим работы для команды (режим «Отладка»).	Проверьте режим работы контроллера.
16#00002205	Неверный режим работы для команды (режим «Мониторинг»).	Проверьте режим работы контроллера.
16#00002206	Неверный режим работы для команды (режим «Выполнение»).	Проверьте режим работы контроллера.
16#00002207	Указанный узел не является опрашивающим узлом.	Проверьте, какой из узлов сети является опрашивающим узлом.

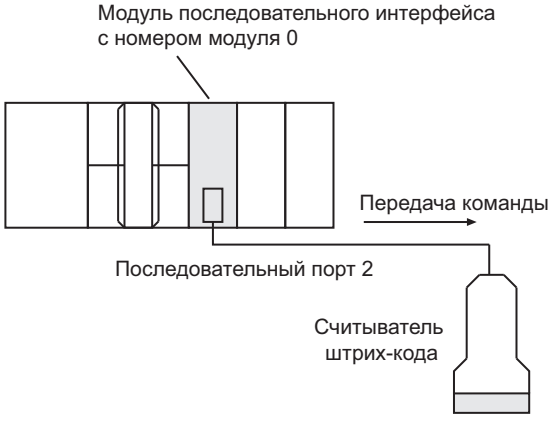
Значение	Ошибка	Способ устранения
16#00002208	Неверный режим работы для команды.	Проверьте состояние активации шага.
16#00002211	Модуль занят.	Увеличьте количество повторных попыток или скорректируйте систему, чтобы коммуникационный трафик не концентрировался на соответствующем модуле.
16#00002301	Файлового устройства не существует.	Вставьте носитель. Или отформатируйте память EM.
16#00002302	Файловой памяти нет.	Проверьте устройство файловой памяти.
16#00002303	Встроенных часов нет.	Проверьте технические характеристики модели.
16#00002401	Ошибка контрольной суммы в данных макроса протокола или передача данных еще не завершена.	Передайте данные макроса протокола из CX-Protocol еще раз.
16#00002502	Имеется ошибка в памяти.	Передайте в память правильные данные.
16#00002503	Зарегистрированная конфигурация модулей ввода-вывода не соответствует фактической конфигурации модулей.	Проверьте конфигурацию модулей ввода-вывода.
16#00002504	Слишком много локальных или удаленных точек ввода-вывода.	Правильно задайте количество локальных и удаленных точек ввода-вывода.
16#00002505	Произошла ошибка при обмене данными между модулем ЦПУ и модулем шины ЦПУ.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#00002506	Один и тот же номер стойки, номер модуля или адрес ввода/вывода задан более одного раза.	Исправьте настройки, чтобы каждый номер был уникальным.
16#00002507	Произошла ошибка при обмене данными между модулем ЦПУ и модулем ввода-вывода.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#00002509	Ошибка передачи данных в SYSMAC BUS/2.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#0000250A	Произошла ошибка при передаче данных модуля шины ЦПУ.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#0000250D	Один и тот же канал используется более одного раза.	Правильно настройте каналы ввода-вывода.
16#0000250F	Имеется ошибка в памяти.	• В случае внутренней памяти: запишите правильные данные и выполните команду еще раз. • В случае карты памяти или файловой памяти EM: выполните команду форматирования памяти расширения. • Если указанные выше меры не помогают устранить ошибку, замените память.
16#00002510	Неверно настроена конечная станция.	Правильно задайте конечную станцию.
16#00002601	Защита уже снята.	Снимать защиту не требуется.
16#00002602	Введен неверный пароль.	Укажите правильный пароль.
16#00002604	Данные защищены.	• Снимите защиту от записи и выполните команду еще раз. • Дождитесь завершения службы, которая выполняется в данный момент, или остановите службу и снова выполните команду.

Значение	Ошибка	Способ устранения
16#00002605	Служба занята.	Дождитесь завершения службы, которая выполняется в данный момент, или остановите службу и снова выполните команду.
16#00002606	Служба остановлена.	При необходимости выполните соответствующую службу.
16#00002607	У вас нет права на выполнение.	• Выполните операцию с узла, с которого был произведен доступ к каналу данных. • Если ошибка все еще присутствует после перезапуска, замените контроллер.
16#00002608	Окружение не настроено.	Выполните необходимую настройку.
16#00002609	Необходимые параметры не заданы.	Задайте необходимые параметры.
16#0000260A	Указанный номер уже определен.	Укажите номер действия или номер перехода, который еще не зарегистрирован, и выполните команду еще раз.
16#0000260B	Невозможно сбросить ошибку.	Устраните причину ошибки, а затем выполните команду сброса ошибки.
16#00003001	У вас нет прав доступа.	Подождите, пока будет разрешен доступ, а затем снова выполните команду.
16#00004001	Служба была прервана.	Устраните причину, по которой была прервана служба, и выполните команду еще раз.

Примечание Помимо кодов, приведенных в таблице выше, о наличии ошибки могут свидетельствовать биты 6, 7 и 15 в коде завершения, которые принимают значение ИСТИНА в случае ошибки. Если значение бита 6 или 7 = ИСТИНА, значит имеется ошибка в модуле ЦПУ в месте назначения. Если значение бита 15 = ИСТИНА, значит произошла ошибка при промежуточной передаче данных по сети.

Пример программы

В качестве примера рассмотрим программу, которая передает команду связи в беспротокольном режиме считывателю штрих-кодов, подключенному к последовательному порту 2 модуля последовательного интерфейса серии CJ (номер модуля 0, название устройства «Barcode»).
Передается команда чтения номера режима съемки (@READ).
Передаются данные, содержащиеся в переменной-массиве SendDat[]. Код начала данных отсутствует, используется код конца данных 16#OD (CR).

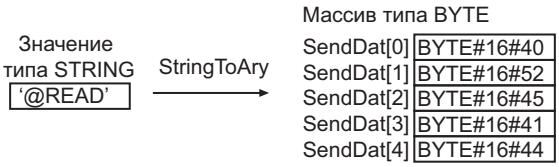


В следующей таблице приведены значения параметров модуля последовательного интерфейса.

Параметр	Заданное значение
Порт 2: включение параметров, задаваемых пользователем	Пользовательские параметры

Параметр	Заданное значение
Порт 2: режим связи по последовательному интерфейсу	Беспротокольный режим (No-protocol)
Порт 2: длина данных	8 бит
Порт 2: количество стоп-битов	1 бит
Порт 2: проверка четности	Нет
Порт 2: скорость передачи данных	38 400 бит/с
Порт 2: код конца данных в беспротокольном режиме	D
Порт 2: настройка включения кода начала данных в беспротокольном режиме	Нет
Порт 2: настройка включения кода конца данных в беспротокольном режиме	Да (укажите требуемый код конца данных)

Текстовая строка «@READ» разбивается на отдельные символы, коды символов хранятся в элементах массива SendDat[]. В элементе SendDat[0] хранится значение BYTE#16#40 (@), в элементе SendData[1] — значение BYTE#16#52(R) и т. д. Для сохранения кодов символов в массив используется команда StringToAry.



Определения глобальных переменных

● Глобальные переменные

Имя	Тип данных	Параметр «AT»*1	Комментарий
Barcode_P2_NopSerialSendExecSta	BOOL	IOBus://rack#0/slot#0/P2_NopSta/P2_NopSerialSendExecSta	Флаг выполнения команды SerialSend

*1. Параметр «AT» для случая, когда модуль последовательного интерфейса установлен в слот с номером 0 стойки с номером 0.

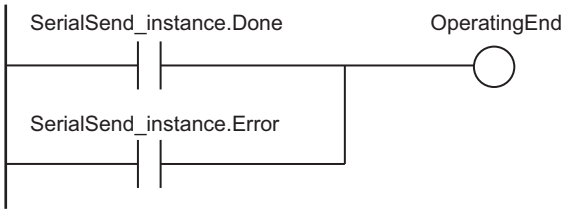
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	InPort	_sPORT	(UnitNo:=_CBU_No00, PhysicPortNo:=0)	Параметры порта
	SendDat	ARRAY[0..4] OF BYTE	[5(16#0)]	Передаваемые данные

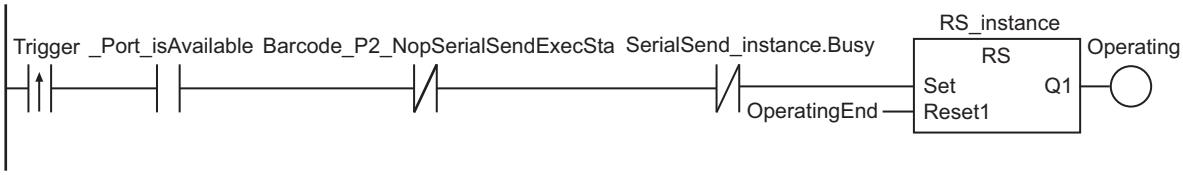
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	RS_instance	RS		
	SerialSend_instance	SerialSend		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Port_isAvailable	BOOL	Флаг активации команд сетевой связи
	Barcode_P2_NopSerialSendExecSta	BOOL	Флаг выполнения команды SerialSend

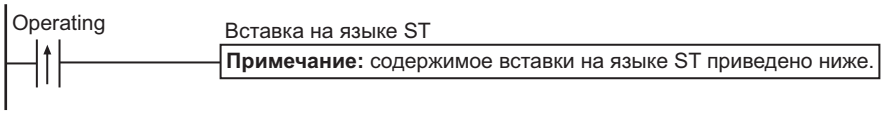
Проверка завершения выполнения команды SerialSend.



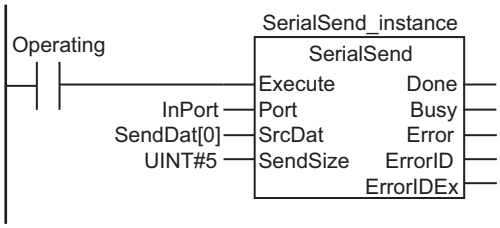
Прием условия выполнения.



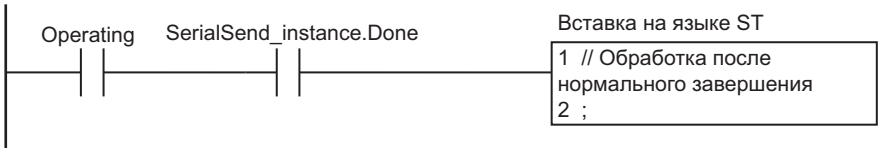
Настройка параметров связи.

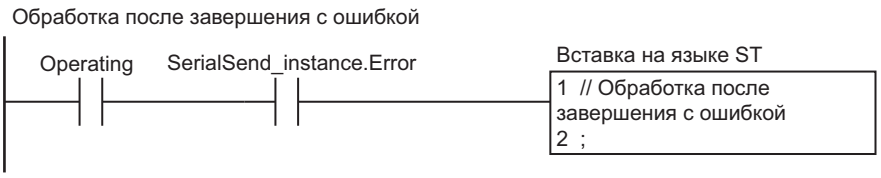


Выполнение команды SerialSend.



Обработка после нормального завершения





● Содержимое вставки на языке ST

```
StringToAry(In:='@READ', AryOut:=SendDat[0]);    // Подготовка SendDat[].
InPort.UnitNo      :=_CBU_No00;                  // Модуль послед. интерф. с номером
 модуля 0
InPort.PhysicPortNo:=USINT#2;                    // Последовательный порт 2
```

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась
	Operating	BOOL	ЛОЖЬ	Обработка
	InPort	_sPORT	(UnitNo:=_CBU_No00 , PhysicPortNo:=0)	Параметры порта
	SendDat	ARRAY[0..4] OF BYTE	[5(16#0)]	Передаваемые данные
	SerialSend_instance	SerialSend		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Port_isAvailable	BOOL	Флаг активации команд сетевой связи
	Barcode_P2_NopSerialSendExecSta	BOOL	Флаг выполнения команды SerialSend

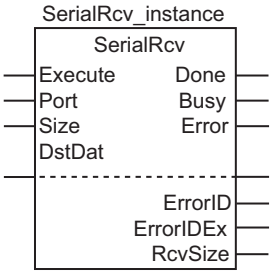
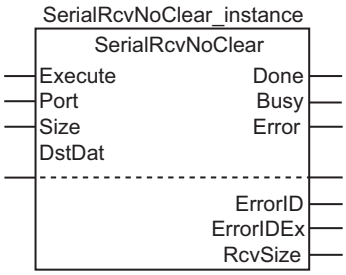
```
// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Port_isAvailable=TRUE)
      AND (Barcode_P2_NopSerialSendExecSta=FALSE) AND (SerialSend_instance.Busy=FAL
SE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
```

```
LastTrigger:=Trigger;

// Настройка параметров связи и инициализация команды SerialSend.
IF (OperatingStart=TRUE) THEN
    SerialSend_instance(
        Execute:=FALSE,
        SrcDat :=SendDat[0]);
    StringToAry(In:='@READ', AryOut:=SendDat[0]);
    InPort.UnitNo      :=_CBU_No00;          // Модуль послед. интерф. с номером мо
дуля 0
    InPort.PhysicPortNo:=USINT#2;            // Последовательный порт 2
    OperatingStart     :=FALSE;
END_IF;

// Выполнение команды SerialSend.
IF (Operating=TRUE) THEN
    SerialSend_instance(
        Execute :=TRUE,
        Port    :=InPort,      // Параметры порта
        SrcDat  :=SendDat[0],  // Передаваемые данные
        SendSize:=UINT#5);     // Объем передаваемых данных

    IF (SerialSend_instance.Done=TRUE) THEN
        // Обработка после нормального завершения
        Operating:=FALSE;
    END_IF;

    IF (SerialSend_instance.Error=TRUE) THEN
        // Обработка после завершения с ошибкой
        Operating:=FALSE;
   END_IF;
END_IF;
```

SerialRcv и SerialRcvNoClear

Команды SerialRcv и SerialRcvNoClear выполняют прием данных в беспротокольном режиме через последовательный порт модуля последовательного интерфейса.

SerialRcv : Очищает буфер приема после чтения данных.
SerialRcvNoClear : Не очищает буфер приема после чтения данных.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SerialRcv	Прием через последовательный порт SCU	FB		SerialRcv_instance(Execute, Port, Size, DstDat, Done, Busy, Error, ErrorID, ErrorIDEx, RcvSize);
SerialRcvNoClear	Прием через последовательный порт SCU без очистки буфера приема	FB		SerialRcvNoClear_instance(Execute, Port, Size, DstDat, Done, Busy, Error, ErrorID, ErrorIDEx, RcvSize);

Меры предосторожности для обеспечения надлежащей эксплуатации

Эти команды нельзя использовать с модулями ЦПУ серии NX.

Информация о версии

Для использования команды SerialRcvNoClear требуются модуль ЦПУ с версией модуля не ниже 1.03, Sysmac Studio версии 1.04 или выше и модуль последовательного интерфейса с версией модуля не ниже 2.1.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Port	Порт назначения	Вход	Порт назначения	---	---	---
Size	Объем принятых данных		Объем принятых данных, сохраняемых в DstDat[]	0...256	Байты	1
DstDat[] (массив)	Массив принимаемых данных	Вход-выход	Массив принимаемых данных	Зависит от типа данных.	---	---

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
RcvSize	Объем сохраненных принятых данных	Выход	Объем принятых данных, фактически сохраненных в массив DstDat[]	0...256	Байты	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Port	Подробные сведения о структуре _sPORT см. в разделе *Тип данных переменной Port (порт назначения)* на стр. 2-1405.																			
Size							ОК													
DstDat[] (массив)		ОК																		
RcvSize							ОК													

Функция

Сначала данные, принятые в беспротокольном режиме (No-protocol) через последовательный порт, указанный параметром *Port*, сохраняются в буфер приема в модуле последовательного интерфейса.

Команды SerialRcv и SerialRcvNoClear передают данные, объем которых указан параметром *Size* (объем принятых данных), из буфера приема в массив DstDat[] (массив принимаемых данных).

После того как данные переданы, в переменную *RcvSize* (объем сохраненных принятых данных) записывается фактическое количество элементов массива DstDat[], в которые были сохранены данные.

Если объем данных в буфере приема меньше, чем значение *Size*, в массив DstDat[] передаются все данные, содержащиеся в буфере приема. В переменную *RcvSize* записывается фактический объем данных, переданных и сохраненных в массив DstDat[].

Когда объем данных в буфере приема равен или превышает ***Size***

Когда объем данных в буфере приема меньше значения ***Size***

Код начала данных и код конца данных в принимаемых данных

Для распознавания кодов начала и конца данных в принимаемых данных в программе пользователя используются переменные устройства. Коды начала и конца данных удаляются из принятых данных перед их сохранением в массив DstDat[].

Присоединяемый код	Переменная устройства для порта 1	Значение
Указанный код начала данных	P1_NopStartCodeYNCfg	ИСТИНА
	P1_NopStartCodeCfg	Код начала данных (16#00...16#FF)
Указанный код конца данных	P1_NopEndCodeYNCfg	ИСТИНА
	P1_NopCRLFCfg	ЛОЖЬ
	P1_NopEndCodeCfg	Код конца данных (16#00...16#FF)
CR+LF в качестве кода конца данных	P1_NopEndCodeYNCfg	ИСТИНА
	P1_NopCRLFCfg	ИСТИНА

В случае добавления кодов начала и конца данных максимальное количество принимаемых байтов составляет 259 (1 байт кода начала данных, 2 байта кода конца данных (для указания кода CR+LF) и 256 байтов передаваемых данных).

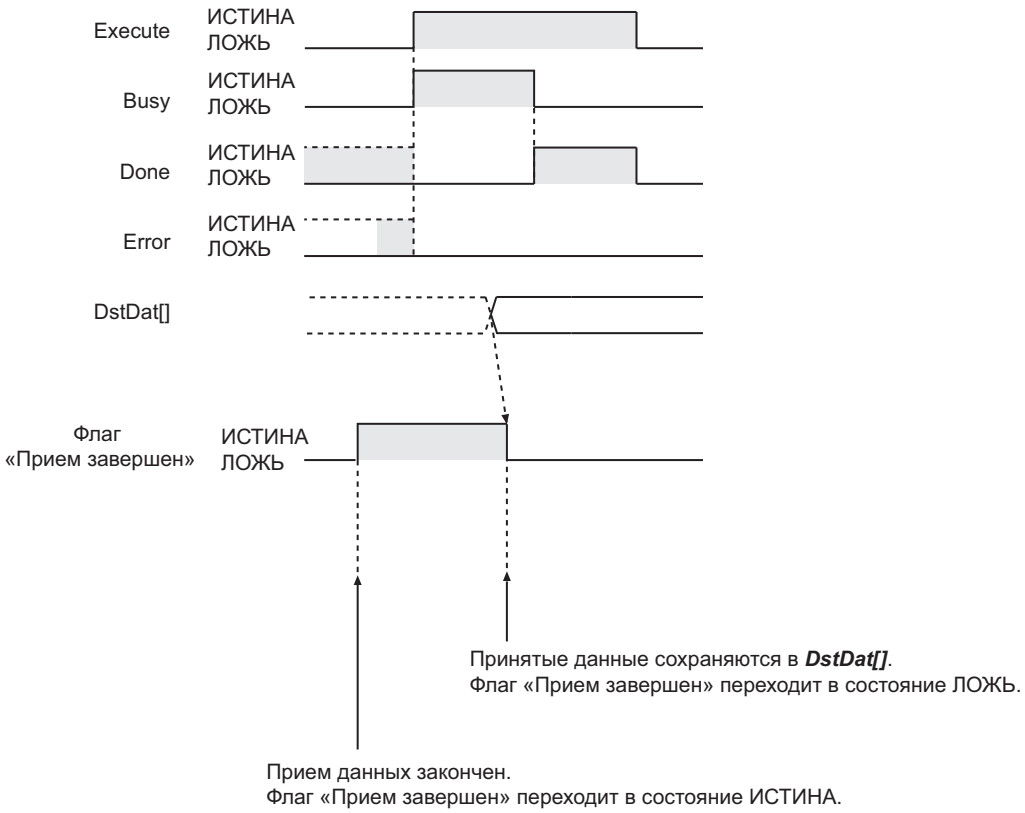
Тип данных переменной *Port* (порт назначения)

Для переменной *Port* (порт назначения) используется структурный тип данных _sPORT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Port	Порт назначения	Порт назначения	_sPORT	---	---	---
UnitNo	Номер модуля	Номер модуля для модуля последовательного интерфейса	_eUnitNo	_CBU_No00... _CBU_No15	---	_CBU_No00
PhysicPortNo	Номер последовательного порта	Номер последовательного порта на модуле последовательного интерфейса	USINT	1 или 2		1

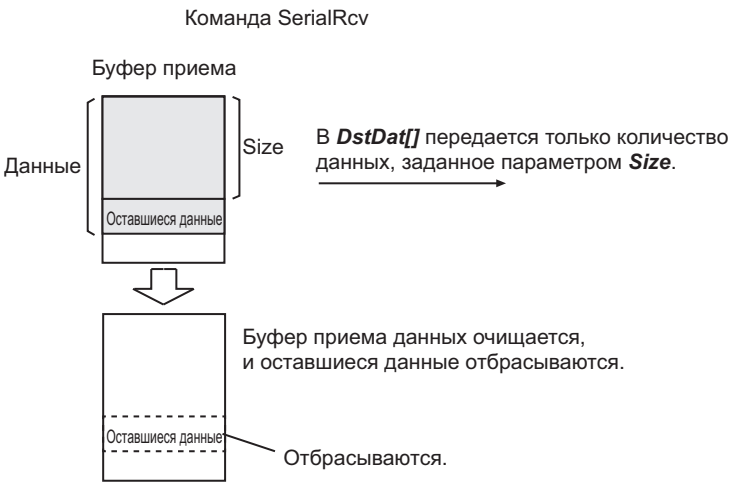
Временная диаграмма

На рисунке ниже представлена временная диаграмма, демонстрирующая работу команды.

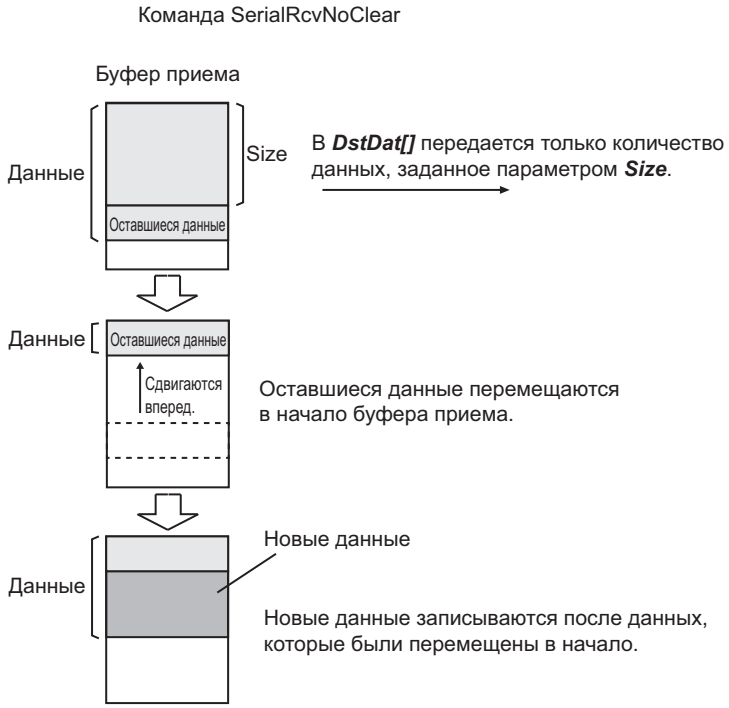


Различия между командами SerialRcv и SerialRcvNoClear

Команды SerialRcv и SerialRcvNoClear отличаются друг от друга тем, как они поступают с данными в буфере приема после того, как данные переданы из буфера приема в массив DstDat[]. Команда SerialRcv очищает буфер приема после того, как она передала данные. Следовательно, если объем данных в буфере приема больше, чем значение *Size*, «лишние» данные останутся в буфере и будут удалены после завершения передачи.



Команда SerialRcvNoClear удаляет только переданные данные. Данные, оставшиеся в буфере приема, перемещаются в его начало. Если после этого в буфер приема поступают новые данные, они записываются после данных, перемещенных в начало буфера.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Port_numUsingPort	Количество используемых портов	USINT	Количество портов, которые используются в настоящее время.
_Port_isAvailable	Флаг активации команд сетевой связи	BOOL	ИСТИНА: порт доступен. ЛОЖЬ: порт недоступен.

Связанные переменные, частично определяемые пользователем

Имя	Значение	Тип данных	Описание
P#_NopRcvOvfSta*1	Флаг переполнения при приеме данных	BOOL	ИСТИНА: объем данных, принятых модулем, превысил указанный объем данных (т. е. были приняты данные после перехода флага завершения приема в состояние ИСТИНА). ЛОЖЬ: объем данных, принятых модулем, не превысил указанный объем данных.
P#_NopRcvCompleteSta*1	Флаг завершения приема	BOOL	ИСТИНА: прием был завершен. ЛОЖЬ: данные не были приняты или принимаются в данный момент.
P#_NopRcvCntSta*1	Счетчик приема	UINT	16#0000...16#0100: количество байтов принятых данных
P#_NopStartCodeYNCfg*1	Активация кода начала данных в беспротокольном режиме	BOOL	ИСТИНА: использовать код начала данных ЛОЖЬ: нет
P#_NopEndCodeYNCfg*1	Активация кода конца данных в беспротокольном режиме	BOOL	ИСТИНА: использовать код конца данных ЛОЖЬ: Нет

Имя	Значение	Тип данных	Описание
P#_NopCRLFCfg*1	Указание кода CR LF в беспротокольном режиме	BOOL	ИСТИНА: использовать CR+LF ЛОЖЬ: не использовать CR+LF
P#_NopRcvDatSzCfg*1	Количество принимаемых байтов данных в беспротокольном режиме	USINT	16#01...16#FF: 1...255 байт 16#00: 256 байт
P#_NopStartCodeCfg*1	Код начала данных в беспротокольном режиме	USINT	16#00...16#FF
P#_NopEndCodeCfg*1	Код конца данных в беспротокольном режиме	USINT	16#00...16#FF
P#_TransErr*1	Ошибка передачи	BOOL	ИСТИНА: произошла ошибка. ЛОЖЬ: ошибок не произошло.
P#_OverRunErr*1	Ошибка переполнения	BOOL	ИСТИНА: произошла ошибка. ЛОЖЬ: ошибок не произошло.

*1. "#" — следует подставить номер порта модуля последовательного интерфейса.

Дополнительная информация

- Флаг *P#_NopRcvCompleteSta* (флаг завершения приема) переходит в состояние ИСТИНА, когда происходит следующее:
  a) Объем принятых данных достиг значения, указанного в *P#_NopRcvDatSzCfg* (количество принимаемых байтов данных в беспротокольном режиме).
  b) Принят указанный код конца данных.
  c) Принято 256 байтов данных.
- Флаг *P#_NopRcvOvfSta* (флаг переполнения при приеме) переходит в состояние ИСТИНА, когда происходит следующее:
  a) После перехода флага *P#_NopRcvCompleteSta* (флаг завершения приема) в состояние ИСТИНА были приняты дополнительные данные до выполнения команды SerialRcv или SerialRcvNoClear.
  b) Объем принятых данных превысил значение, указанное в *P#_NopRcvDatSzCfg* (количество принимаемых байтов данных в беспротокольном режиме).
- Дополнительные сведения о передаче данных в беспротокольном режиме см. в руководстве *Серия CJ, модули последовательного интерфейса — Использование с модулем ЦПУ серии NJ. Руководство по работе (Cat. No. W494)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эти команды следует выполнять, когда флаг *P#_NopRcvCompleteSta* (флаг завершения приема) = ИСТИНА.
- После того как данные приняты, следует всегда выполнять эту команду для передачи данных из буфера приема в массив DstDat[]. Пока предыдущие данные не будут переданы в массив, прием новых данных будет невозможен.
- После того как объем принятых данных достигает 259 байт, прием новых данных автоматически прекращается. Если до выполнения команды SerialRcv или SerialRcvNoClear оказываются

приняты дополнительные данные, флаг *P#_OverRunErr* (ошибка переполнения) переходит в состояние ИСТИНА.

- При добавлении кода начала или кода конца данных его не следует учитывать в значении *Size*.
- Эту команду можно использовать только для последовательного порта модуля последовательного интерфейса, который был переведен в режим «Без протокола».
- Если значение *Size* = 0, содержимое буфера приема не будет передано в массив DstDat[]. В этом случае флаги *P#_NopRcvCompleteSta* (флаг завершения приема) и *P#_NopRcvOvfSta* (флаг переполнения при приеме данных) перейдут в состояние ЛОЖЬ. А значение переменной *P#_NopRcvCntSta* (счетчик приема) будет равно 0.
- Команда выполняется только при наличии доступного порта. Поэтому системную переменную *_Port_isAvailable* (флаг активации команд сетевой связи) следует использовать для команды в качестве нормально разомкнутого условия выполнения.
- Команда не выполняется, пока переменная *Busy* = ИСТИНА. Поэтому переменную *Busy* следует использовать для команды в качестве нормально замкнутого условия выполнения.
- Если команда используется в программе на языке ST, необходимо обеспечить, чтобы команда обрабатывалась в каждом цикле выполнения задачи в течение всего времени ее выполнения. В противном случае нормальная обработка может оказаться невозможной.
- При выполнении команды SerialRcv производится очистка буфера приема в модуле последовательного интерфейса. Поэтому данные в буфере приема нельзя разбить на отдельные блоки для передачи в массив DstDat[].
- Как и в случае команды SerialRcv, если объем принятых данных превышает значение переменной *Size*, «лишние» данные будут отброшены при выполнении другой команды SerialRcv.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) При выполнении команды в качестве режима последовательной связи не установлен режим «Без протокола».
  b) Значение переменной *_Port_isAvailable* = ЛОЖЬ.
  c) Значение *Port.UnitNo* или *Port.PhysicPortNo* вне допустимого диапазона.
  d) Модуль последовательного интерфейса серии CJ с указанным номером модуля отсутствует.
  e) Значение *Size* находится за пределами допустимого диапазона.
  f) Значение *Size* превосходит размер массива DstDat[].
  g) Сбой связи.
  h) Команда выполнена во время перезапуска модуля.
- Для этой команды в качестве дополнительного кода ошибки в переменной *ErrorIDEx* указывается код ответа интерфейса связи.
  В следующей таблице приведены возможные значения кода ошибки и поясняется, что они означают. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = WORD#16#0800.

Значение	Ошибка	Способ устранения
16#00000001	Обслуживание связи было прервано.	• Проверьте состояние работы логических связей. • Проверьте емкость адресуемой области памяти, в которую передаются данные, на стороннем узле.
16#00000101	Локальный узел не является частью сети.	Сделайте локальный узел частью сети.
16#00000102	Превышено время ожидания маркера.	Задайте адрес локального узла так, чтобы он не превышал максимальный адрес узла.
16#00000103	Превышено максимальное количество повторных попыток.	Протестируйте связь между узлами. При обнаружении какой-либо ошибки проверьте программную и аппаратную инфраструктуру.

Значение	Ошибка	Способ устранения
16#00000104	Превышено допустимое количество кадров передачи.	Проверьте состояние событий в сети и уменьшите число событий, приходящихся на каждый цикл выполнения задачи. Либо увеличьте допустимое количество кадров передачи.
16#00000105	IP-адрес локального узла вне допустимого диапазона.	Правильно настройте поворотные переключатели на модуле последовательного интерфейса.
16#00000106	IP-адрес локального узла также используется другим узлом в сети.	Измените дублирующийся адрес на одном из узлов.
16#00000201	Удаленный узел не является частью сети.	Сделайте удаленный узел частью сети.
16#00000202	Модуля с указанным адресом модуля не существует в месте назначения.	Правильно задайте адрес модуля для адреса сети назначения.
16#00000203	Сторонний узел не является частью сети.	• Проверьте адрес модуля, который является сторонним узлом. • Укажите только один узел в качестве стороннего узла.
16#00000204	Удаленный узел занят.	Увеличьте количество повторных попыток или скорректируйте систему, чтобы коммуникационный трафик не концентрировался на удаленном узле.
16#00000205	Превышено время ожидания ответа.	Проверьте настройку параметров связи.
16#00000206	Имеется ошибка в канале передачи данных.	• Выполните операцию повторно. • Если эта ошибка возникает часто, проверьте наличие помех.
16#00000301	Произошла ошибка контроллера связи.	См. руководство по эксплуатации для соответствующего модуля и внесите необходимые исправления.
16#00000302	Ошибка модуля ЦПУ на удаленном узле.	См. документацию по модулю ЦПУ на удаленном узле и устраните ошибку.
16#00000303	В соответствующем контроллере произошла ошибка, и ответ не был возвращен.	Проверьте состояние связи в сети и перезапустите соответствующий контроллер. Если ошибка не исчезла, замените соответствующий контроллер.
16#00000304	Неверно задан номер модуля.	Правильно настройте поворотные переключатели на модуле последовательного интерфейса.
16#00000401	Переданная команда не поддерживается.	Правильно настройте массив команды.
16#00000402	Модель или версия модуля не поддерживаются.	Проверьте модель и версию модуля.
16#00000501	Неверно настроен удаленный адрес.	Задайте адрес назначения в таблице маршрутизации.
16#00000502	Таблицы маршрутизации не зарегистрированы.	Задайте исходный узел, узел назначение и промежуточные узлы в таблицах маршрутизации.
16#00000503	Имеется ошибка в таблицах маршрутизации.	Исправьте настройки в таблицах маршрутизации.
16#00000504	Слишком много промежуточных точек.	Измените конструкцию сети или исправьте таблицы маршрутизации таким образом, чтобы команды передавалась в пределах трех уровней сети.
16#00001001	Слишком длинная команда.	Правильно настройте массив команды.
16#00001002	Слишком короткая команда.	Правильно настройте массив команды.
16#00001003	Количество записываемых элементов, указанное в команде, не согласуется с объемом записываемых данных.	Задайте одинаковое количество записываемых элементов и записываемых данных.
16#00001004	Неверный формат команды.	Правильно настройте массив команды.

Значение	Ошибка	Способ устранения
16#00001005	Имеется ошибка в заголовке.	Исправьте настройки в таблицах маршрутизации.
16#00001101	Тип области не существует.	См. информацию о переменных команды и кодах типов параметров и задайте подходящие коды.
16#00001102	Неверный размер для доступа к данным.	Правильно задайте размеры данных для доступа к переменным и параметрам.
16#00001103	Указан адрес вне допустимого диапазона.	Укажите адрес в пределах допустимого диапазона.
16#00001104	Превышен диапазон адресов.	• Укажите адрес в пределах допустимого диапазона. • Исправьте настройки в таблице логических связей.
16#00001106	Указан незарегистрированный номер коммуникационной последовательности.	Исправьте номер коммуникационной последовательности или добавьте последовательность с помощью CX-Protocol.
16#00001109	Произошла ошибка взаимосвязи.	• Устраните несоответствие размеров друг другу в данных команды. • Исправьте настройки в таблице логических связей.
16#0000110A	Данные избыточны.	• Отмените текущий процесс или дождитесь его завершения, прежде чем выполнять команду. • Исправьте настройки в таблице логических связей.
16#0000110B	Слишком длинный ответ.	Правильно задайте количество элементов в массиве команды.
16#0000110C	Другая ошибка параметров.	Правильно настройте массив команды.
16#00002002	Данные защищены.	Снимите защиту и выполните команду еще раз.
16#00002003	Ни одна таблица не зарегистрирована.	Правильно настройте таблицу.
16#00002004	Нет данных, соответствующих искомым данным.	Правильно настройте данные для поиска.
16#00002005	Соответствующего номера программы не существует.	Задайте действительный номер программы.
16#00002006	Соответствующего файла не существует.	Правильно задайте имя файла, включая имена подкаталогов.
16#00002007	Произошла ошибка проверки.	• Проверьте содержимое памяти и запишите правильные данные. • Проверьте содержимое файла.
16#00002101	Доступ невозможен, так как область предназначена только для чтения.	Снимите защиту от записи и выполните команду еще раз.
16#00002102	Данные защищены или запись в таблицу логических связей невозможна.	• Снимите защиту от записи и выполните команду еще раз. • Задайте системные параметры в таблице логических связей.
16#00002103	Регистрация невозможна.	• Создайте файл после удаления ненужных файлов или подготовьте новую файловую память. • Закройте открытые файлы и выполните команду еще раз.
16#00002105	Соответствующего номера программы не существует.	Задайте действительный номер программы.
16#00002106	Соответствующего файла не существует.	Правильно задайте имя файла, включая имена подкаталогов.
16#00002107	Файл с таким именем уже существует.	Измените имя файла для записи и выполните команду еще раз.
16#00002108	Изменение недопустимо, так как оно вызывает ошибку.	Скорректируйте настройки.
16#00002201	Операция невозможна, так как макрос протокола уже выполняется.	Используйте флаг выполнения макроса протокола в качестве нормально замкнутого входа программы.

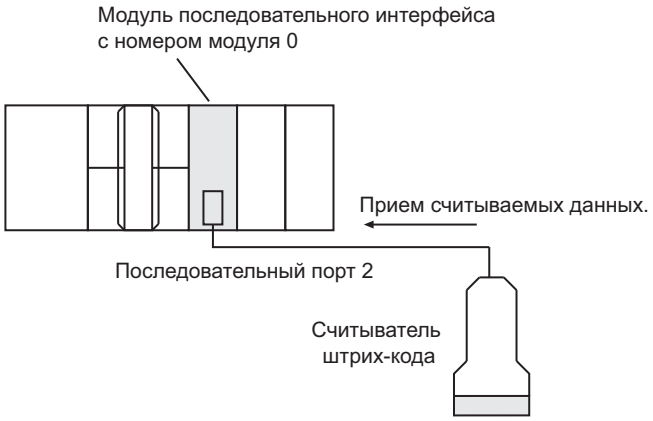
Значение	Ошибка	Способ устранения
16#00002202	Неверный режим работы.	Проверьте режим работы.
16#00002203	Неверный режим работы для команды (режим «Программирование»).	Проверьте режим работы контроллера.
16#00002204	Неверный режим работы для команды (режим «Отладка»).	Проверьте режим работы контроллера.
16#00002205	Неверный режим работы для команды (режим «Мониторинг»).	Проверьте режим работы контроллера.
16#00002206	Неверный режим работы для команды (режим «Выполнение»).	Проверьте режим работы контроллера.
16#00002207	Указанный узел не является опрашивающим узлом.	Проверьте, какой из узлов сети является опрашивающим узлом.
16#00002208	Неверный режим работы для команды.	Проверьте состояние активации шага.
16#00002211	Модуль занят.	Увеличьте количество повторных попыток или скорректируйте систему, чтобы коммуникационный трафик не концентрировался на соответствующем модуле.
16#00002301	Файлового устройства не существует.	Вставьте носитель. Или отформатируйте память EM.
16#00002302	Файловой памяти нет.	Проверьте устройство файловой памяти.
16#00002303	Встроенных часов нет.	Проверьте технические характеристики модели.
16#00002401	Ошибка контрольной суммы в данных макроса протокола или передача данных еще не завершена.	Передайте данные макроса протокола из CX-Protocol еще раз.
16#00002502	Имеется ошибка в памяти.	Передайте в память правильные данные.
16#00002503	Зарегистрированная конфигурация модулей ввода-вывода не соответствует фактической конфигурации модулей.	Проверьте конфигурацию модулей ввода-вывода.
16#00002504	Слишком много локальных или удаленных точек ввода-вывода.	Правильно задайте количество локальных и удаленных точек ввода-вывода.
16#00002505	Произошла ошибка при обмене данными между модулем ЦПУ и модулем шины ЦПУ.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#00002506	Один и тот же номер стойки, номер модуля или адрес ввода/вывода задан более одного раза.	Исправьте настройки, чтобы каждый номер был уникальным.
16#00002507	Произошла ошибка при обмене данными между модулем ЦПУ и модулем ввода-вывода.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#00002509	Ошибка передачи данных в SYSMAC BUS/2.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#0000250A	Произошла ошибка при передаче данных модуля шины ЦПУ.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#0000250D	Один и тот же канал используется более одного раза.	Правильно настройте каналы ввода-вывода.

Значение	Ошибка	Способ устранения
16#0000250F	Имеется ошибка в памяти.	• В случае внутренней памяти: запишите правильные данные и выполните команду еще раз. • В случае карты памяти или файловой памяти EM: выполните команду форматирования памяти расширения. • Если указанные выше меры не помогают устранить ошибку, замените память.
16#00002510	Неверно настроена конечная станция.	Правильно задайте конечную станцию.
16#00002601	Защита уже снята.	Снимать защиту не требуется.
16#00002602	Введен неверный пароль.	Укажите правильный пароль.
16#00002604	Данные защищены.	• Снимите защиту от записи и выполните команду еще раз. • Дождитесь завершения службы, которая выполняется в данный момент, или остановите службу и снова выполните команду.
16#00002605	Служба занята.	Дождитесь завершения службы, которая выполняется в данный момент, или остановите службу и снова выполните команду.
16#00002606	Служба остановлена.	При необходимости выполните соответствующую службу.
16#00002607	У вас нет права на выполнение.	• Выполните операцию с узла, с которого был произведен доступ к каналу данных. • Если ошибка все еще присутствует после перезапуска, замените контроллер.
16#00002608	Окружение не настроено.	Выполните необходимую настройку.
16#00002609	Необходимые параметры не заданы.	Задайте необходимые параметры.
16#0000260A	Указанный номер уже определен.	Укажите номер действия или номер перехода, который еще не зарегистрирован, и выполните команду еще раз.
16#0000260B	Невозможно сбросить ошибку.	Устраните причину ошибки, а затем выполните команду сброса ошибки.
16#00003001	У вас нет прав доступа.	Подождите, пока будет разрешен доступ, а затем снова выполните команду.
16#00004001	Служба была прервана.	Устраните причину, по которой была прервана служба, и выполните команду еще раз.

Примечание Помимо кодов, приведенных в таблице выше, о наличии ошибки могут свидетельствовать биты 6, 7 и 15 в коде завершения, которые принимают значение ИСТИНА в случае ошибки. Если значение бита 6 или 7 = ИСТИНА, значит имеется ошибка в модуле ЦПУ в месте назначения. Если значение бита 15 = ИСТИНА, значит произошла ошибка при промежуточной передаче данных по сети.

Пример программы

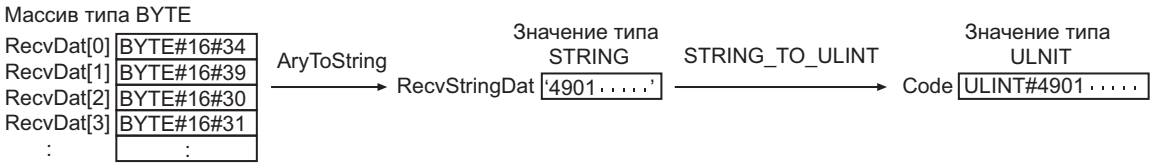
В данном примере производится прием данных, которые были считаны устройством чтения штрих-кодов, подключенным к последовательному порту 2 модуля последовательного интерфейса серии CJ (номер модуля 0, название устройства «Barcode»).
Принятые данные сохраняются в переменную-массив RecvDat[]. Код начала данных отсутствует, используется код конца данных 16#OD (CR).



В следующей таблице приведены значения параметров модуля последовательного интерфейса.

Параметр	Заданное значение
Порт 2: включение параметров, задаваемых пользователем	Пользовательские параметры
Порт 2: режим связи по последовательному интерфейсу	Беспротокольный режим (No-protocol)
Порт 2: длина данных	8 бит
Порт 2: количество стоп-битов	1 бит
Порт 2: проверка четности	Нет
Порт 2: скорость передачи данных	38 400 бит/с
Порт 2: код конца данных в беспротокольном режиме	D
Порт 2: настройка включения кода начала данных в беспротокольном режиме	Нет
Порт 2: настройка включения кода конца данных в беспротокольном режиме	Да (укажите требуемый код конца данных)

Число, получаемое от считывателя штрих-кодов, разбивается на отдельные символы. Битовые строки с кодами символов сохраняются в массив RecvDat[]. Один элемент массива RecvDat[] соответствует одному символу (цифре) штрих-кода. Сначала используется команда AryToString для преобразования данных в текстовую строку (*RecvStringDat*). Затем используется команда STRING_TO_ULINT для преобразования данных в целое число типа ULINT (*Code*).



Определения глобальных переменных

● Глобальные переменные

Имя	Тип данных	Параметр «AT»*1	Комментарий
Barcode_P2_NopRcvCompleteSta	BOOL	IOBus://rack#0/slot#0/P2_NopSta/P2_NopRcvCompleteSta	Флаг завершения приема

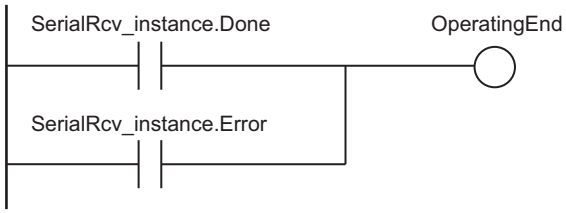
*1. Параметр «AT» для случая, когда модуль последовательного интерфейса установлен в слот с номером 0 стойки с номером 0.

Программа на языке LD

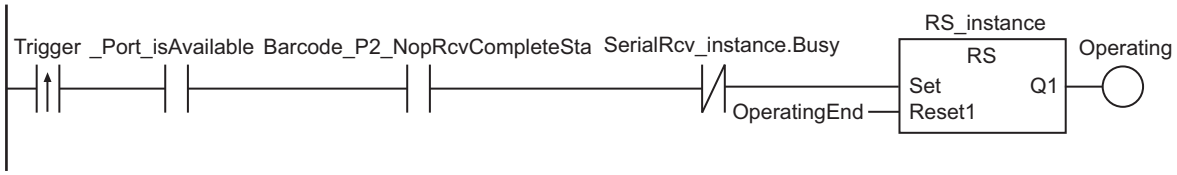
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	InPort	_sPORT	(UnitNo:=_CBU_No00, PhysicPortNo:=0)	Параметры порта
	RecvDat	ARRAY[0..12] OF BYTE	[13(16#0)]	Принимаемые данные
	RecvSize	UINT	0	Объем принятых данных
	RecvStringDat	STRING[255]	''	Значение штрих-кода в виде текстовой строки
	Code	ULINT	0	Значение штрих-кода в виде целого числа
	RS_instance	RS		
	SerialRcv_instance	SerialRcv		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Port_isAvailable	BOOL	Флаг активации команд сетевой связи
	Barcode_P2_NopRcvCompleteSta	BOOL	Флаг завершения приема

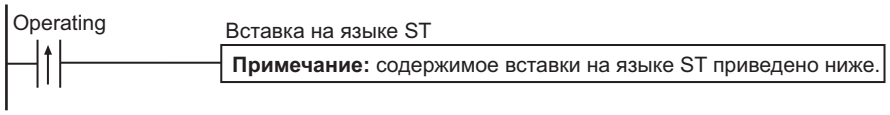
Проверка завершения выполнения команды SerialRcv.



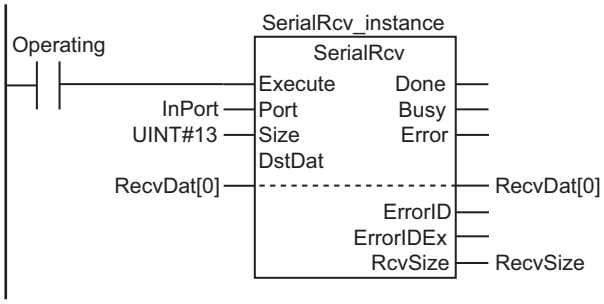
Прием условия выполнения.



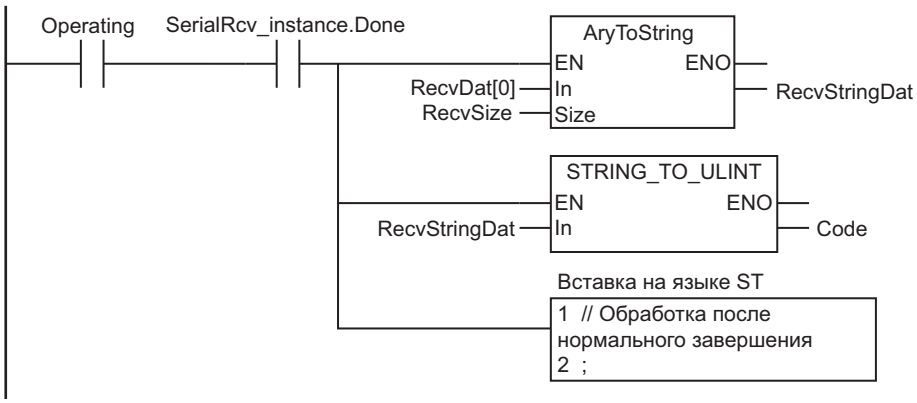
Настройка параметров связи.



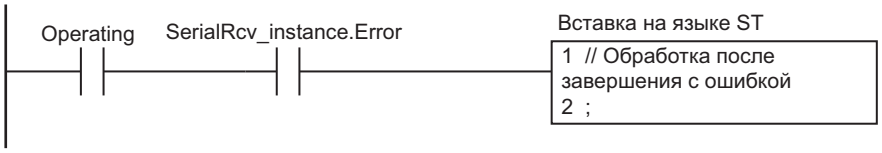
Выполнение команды SerialRcv.



Обработка после нормального завершения



Обработка после завершения с ошибкой



● Содержимое вставки на языке ST

```
InPort.UnitNo      :=_CBU_No00;   // Модуль послед. интерф. с номером модуля 0
InPort.PhysicPortNo:=USINT#2;     // Последовательный порт 2
```

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась
	Operating	BOOL	ЛОЖЬ	Обработка
	InPort	_sPORT	(UnitNo:=_CBU_No00, PhysicPortNo:=0)	Параметры порта

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	RecvDat	ARRAY[0..12] OF BYTE	[13(16#0)]	Принимаемые данные
	RecvSize	UINT	0	Объем принятых данных
	RecvStringDat	STRING[255]	''	Значение штрихкода в виде текстовой строки
	Code	ULINT	0	Значение штрихкода в виде целого числа
	SerialRcv_instance	SerialRcv		

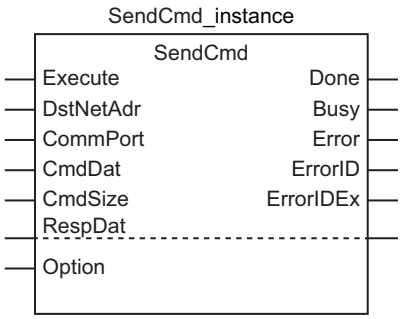
Внешние переменные	Переменная	Тип данных	Комментарий
	_Port_isAvailable	BOOL	Флаг активации команд сетевой связи
	Barcode_P2_NopRcvCompleteSta	BOOL	Флаг завершения приема

```
// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Port_isAvailable=TRUE)
      AND (Barcode_P2_NopRcvCompleteSta=TRUE) AND (SerialRcv_instance.Busy=FALSE) )
 THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Настройка параметров связи и инициализация команды SerialRcv.
IF (OperatingStart=TRUE) THEN
    SerialRcv_instance(
        Execute:=FALSE,      // Инициализация экземпляра.
        Port   :=InPort,     // Параметры порта
        Size   :=UINT#13,    // Объем принимаемых данных
        DstDat :=RecvDat[0], // Принятые данные
        RcvSize =>RecvSize); // Объем фактически принятых данных
    InPort.UnitNo      :=_CBU_No00; // Модуль послед. интерф. с номером модуля 0
    InPort.PhysicPortNo:=USINT#2;   // Последовательный порт 2
    OperatingStart     :=FALSE;
END_IF;

// Выполнение команды SerialRcv.
IF (Operating=TRUE) THEN
    SerialRcv_instance(
        Execute:=TRUE,
        Port   :=InPort,
        Size   :=UINT#13,
        DstDat :=RecvDat[0],
        RcvSize =>RecvSize);
```

```
    IF (SerialRcv_instance.Done=TRUE) THEN
        // Обработка после нормального завершения
        RecvStringDat:=AryToString(In:=RecvDat[0], Size:=RecvSize);  // Преобразова
ние кодов символов в текстовую строку
        Code         :=STRING_TO_ULINT(RecvStringDat);               // Преобразова
ние текстовой строки в целое число.
        Operating    :=FALSE;
    END_IF;
    IF (SerialRcv_instance.Error=TRUE) THEN
        // Обработка после завершения с ошибкой
        Operating:=FALSE;
    END_IF;
END_IF;
```

SendCmd

Команда SendCmd служит для передачи команды модулю последовательного интерфейса с использованием шлюза последовательного интерфейса.
С ее помощью также можно передать в явном виде команду модулю сети DeviceNet или модулю ведущего устройства сети CompoNet.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
SendCmd	Передача команды	FB		SendCmd_instance(Execute, DstNetAdr, CommPort, CmdDat, CmdSize, RespDat, Option, Done, Busy, Error, ErrorID, ErrorIDEx);

Меры предосторожности для обеспечения надлежащей эксплуатации

Данную команду невозможно использовать с модулями ЦПУ серии NX, к которым не могут быть подключены функциональные модули серии CJ.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DstNetAdr	Адрес сети назначения	Вход	Адрес сети назначения	---	---	---
CommPort	Последовательный порт назначения		Последовательный порт назначения	_NONE	---	_NONE
CmdDat[] (массив)	Массив команды		Передаваемая команда	Зависит от типа данных.		*1
CmdSize	Объем данных команды		Объем данных команды	0...макс. длина данных *2	Байты	2
Option	Ответ		Параметры мониторинга ответа и выполнения повторных попыток	---	---	---
RespDat[] (массив)	Массив хранения ответа	Вход-выход	Массив для хранения полученного ответа	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.
*2. Это значение может изменяться в зависимости от типа сети.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DstNetAdr	Подробные сведения о структуре _sDNET_ADR см. в разделе *Функция* на стр. 2-1420.																			
CommPort	Сведения о перечислителе перечислимого типа _ePORT см. в разделе *Функция* на стр. 2-1420.																			
CmdDat[] (массив)		ОК																		
CmdSize							ОК													
Option	Подробные сведения о структуре _sRESPONSE см. в разделе *Функция* на стр. 2-1420.																			
RespDat[] (массив)		ОК																		

Функция

Команда SendCmd передает содержимое массива CmdDat[] (массив команды) адресату, который указывается параметрами *DstNetAdr* (адрес сети назначения) и *CommPort* (последовательный порт назначения).
В параметре *CmdSize* (объем данных команды) указывается количество элементов массива CmdDat[], содержащих данные для передачи команды.
Возвращаемый ответ сохраняется в массив RespDat[] (массив хранения ответа).

Для переменной *DstNetAdr* используется структурный тип данных _sDNET_ADR. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DstNetAdr	Адрес сети назначения	Адрес сети назначения	_sDNET_ADR	---	---	---
NetNo	Адрес сети	Адрес сети	USINT	0...127	---	0
NodeNo	Адрес узла	Адрес узла	USINT	Зависит от типа данных.		
UnitNo	Адрес модуля	Адрес модуля	BYTE	Зависит от типа данных.		16#00

Для переменной *CommPort* используется перечислимый тип данных _ePORT.
Значения перечислителей перечислимого типа _ePORT приведены в таблице ниже.

Перечислители	Значение
_NONE	Портом назначения не является последовательный порт в режиме Host Link.

Для переменной *Option* используется структурный тип данных _sRESPONSE. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Option	Ответ	Параметры мониторинга ответа и выполнения повторных попыток	_sRESPONSE	---	---	---
isNonResp	Без ответа	ИСТИНА: ответ не требуется. ЛОЖЬ: ответ требуется.	BOOL	Зависит от типа данных.	---	ЛОЖЬ
TimeOut	Время ожидания	Время ожидания 0: 2,0 с	UINT		0,1 с	20 (2,0 с)
Retry	Количество повторных попыток	Количество повторных попыток	USINT	0...15	Кол-во раз	0

Если ответ не возвращается в течение заданного времени ожидания (*Option.TimeOut*), когда значение флага «Ответ не требуется» (*Option.isNonResp*) = ЛОЖЬ, команда предпринимает повторные попытки, пока не будет получен ответ.
Количество повторных попыток указывается в переменной *Option.Retry*.
Фактическое время ожидания = *Option.TimeOut* x 0,1 с. Но если значение *Option.TimeOut* = 0, то время ожидания принимается равным 2,0 с. По умолчанию переменная *Option.TimeOut* равна 2,0 с.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Port_numUsingPort	Количество используемых портов	USINT	Количество портов, которые используются в настоящее время.
_Port_isAvailable	Флаг активации команд сетевой связи	BOOL	ИСТИНА: порт доступен. ЛОЖЬ: порт недоступен.

Дополнительная информация

- Команды или возвращаемые ответы в процессе обмена данными могут не доходить до получателей из-за воздействия помех или других факторов. Для повышения надежности связи в переменную *Option.Retry* можно передать значение, отличное от 0, чтобы команда предпринимала повторную попытку в случае отсутствия ответа.
- Сведения о том, как указать адрес сети назначения, см. в документе *Серия SYSMAC CS/CJ/CP/NSJ — Команды связи. Справочное руководство (Cat. No. W342)*.
  У модулей ЦПУ серии CS/CJ и модулей ЦПУ серии NX имеются различия в части спецификаций маршрутизации FINS.
  Информацию о различиях в спецификациях маршрутизации см. в разделе *Отличия между сериями CS/CJ и серией NX в части маршрутизации FINS* документа *Серия NX, модули ЦПУ — Протокол FINS. Руководство пользователя (Cat. No. W596)*.
- Чтобы указать последовательный порт с функцией шлюза последовательного интерфейса, в параметре *DstNetAdr.UnitNo* следует указать адрес модуля последовательного порта. Значения адреса модуля для портов модулей последовательного интерфейса указаны ниже.
  a) Порт 1
  Адрес модуля = BYTE#16#80 + BYTE#16#04 × номер модуля (hex)

Пример для номера модуля 1
BYTE#16#80 + BYTE#16#04 × 1 = BYTE#16#84

b) Порт 2
Адрес модуля = BYTE#16#81 + BYTE#16#04 × номер модуля (hex)
Пример для номера модуля 2
BYTE#16#81 + BYTE#16#04 × 2 = BYTE#16#89

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если *CmdSize* = 0, команда не передается. В этом случае значение переменной *Done* меняется на ИСТИНА при выполнении команды.
- Команда выполняется только при наличии доступного порта. Поэтому системную переменную *_Port_isAvailable* (флаг активации команд сетевой связи) следует использовать для команды в качестве нормально разомкнутого условия выполнения.
- Если команда используется в программе на языке ST, необходимо обеспечить, чтобы команда обрабатывалась в каждом цикле выполнения задачи в течение всего времени ее выполнения. В противном случае нормальная обработка может оказаться невозможной.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Значение переменной *_Port_isAvailable* = ЛОЖЬ.
  b) Значение *CommPort* вне допустимого диапазона.
  c) Значение члена структуры *DstNetAdr* вне допустимого диапазона.
  d) Значение *CmdSize* вне допустимого диапазона.
  e) Значение члена структуры *Option* вне допустимого диапазона.
  f) Значение *CmdSize* превосходит размер массива CmdDat[].
  g) Объем данных ответа превосходит размер массива RespDat[].
  h) Сбой связи.
- Для этой команды в качестве дополнительного кода ошибки в переменной *ErrorIDEx* указывается код ответа интерфейса связи.
  В следующей таблице приведены возможные значения кода ошибки и поясняется, что они означают. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = WORD#16#0800.

Значение	Ошибка	Способ устранения
16#00000001	Обслуживание связи было прервано.	• Проверьте состояние работы логических связей. • Проверьте емкость адресуемой области памяти, в которую передаются данные, на стороннем узле.
16#00000101	Локальный узел не является частью сети.	Сделайте локальный узел частью сети.
16#00000102	Превышено время ожидания маркера.	Задайте адрес локального узла так, чтобы он не превышал максимальный адрес узла.
16#00000103	Превышено максимальное количество повторных попыток.	Протестируйте связь между узлами. При обнаружении какой-либо ошибки проверьте программную и аппаратную инфраструктуру.

Значение	Ошибка	Способ устранения
16#00000104	Превышено допустимое количество кадров передачи.	Проверьте состояние событий в сети и уменьшите число событий, приходящихся на каждый цикл выполнения задачи. Либо увеличьте допустимое количество кадров передачи.
16#00000105	IP-адрес локального узла вне допустимого диапазона.	Правильно настройте поворотные переключатели на модуле последовательного интерфейса.
16#00000106	IP-адрес локального узла также используется другим узлом в сети.	Измените дублирующийся адрес на одном из узлов.
16#00000201	Удаленный узел не является частью сети.	Сделайте удаленный узел частью сети.
16#00000202	Модуля с указанным адресом модуля не существует в месте назначения.	Правильно задайте адрес модуля для адреса сети назначения.
16#00000203	Сторонний узел не является частью сети.	• Проверьте адрес модуля, который является сторонним узлом. • Укажите только один узел в качестве стороннего узла.
16#00000204	Удаленный узел занят.	Увеличьте количество повторных попыток или скорректируйте систему, чтобы коммуникационный трафик не концентрировался на удаленном узле.
16#00000205	Превышено время ожидания ответа.	Проверьте настройку параметров связи.
16#00000206	Имеется ошибка в канале передачи данных.	• Выполните операцию повторно. • Если эта ошибка возникает часто, проверьте наличие помех.
16#00000301	Произошла ошибка контроллера связи.	См. руководство по эксплуатации для соответствующего модуля и внесите необходимые исправления.
16#00000302	Ошибка модуля ЦПУ на удаленном узле.	См. документацию по модулю ЦПУ на удаленном узле и устраните ошибку.
16#00000303	В соответствующем контроллере произошла ошибка, и ответ не был возвращен.	Проверьте состояние связи в сети и перезапустите соответствующий контроллер. Если ошибка не исчезла, замените соответствующий контроллер.
16#00000304	Неверно задан номер модуля.	Правильно настройте поворотные переключатели на модуле последовательного интерфейса.
16#00000401	Переданная команда не поддерживается.	Правильно настройте массив команды.
16#00000402	Модель или версия модуля не поддерживаются.	Проверьте модель и версию модуля.
16#00000501	Неверно настроен удаленный адрес.	Задайте адрес назначения в таблице маршрутизации.
16#00000502	Таблицы маршрутизации не зарегистрированы.	Задайте исходный узел, узел назначение и промежуточные узлы в таблицах маршрутизации.
16#00000503	Имеется ошибка в таблицах маршрутизации.	Исправьте настройки в таблицах маршрутизации.
16#00000504	Слишком много промежуточных точек.	Измените конструкцию сети или исправьте таблицы маршрутизации таким образом, чтобы команды передавалась в пределах трех уровней сети.
16#00001001	Слишком длинная команда.	Правильно настройте массив команды.
16#00001002	Слишком короткая команда.	Правильно настройте массив команды.
16#00001003	Количество записываемых элементов, указанное в команде, не согласуется с объемом записываемых данных.	Задайте одинаковое количество записываемых элементов и записываемых данных.
16#00001004	Неверный формат команды.	Правильно настройте массив команды.

Значение	Ошибка	Способ устранения
16#00001005	Имеется ошибка в заголовке.	Исправьте настройки в таблицах маршрутизации.
16#00001101	Тип области не существует.	См. информацию о переменных команды и кодах типов параметров и задайте подходящие коды.
16#00001102	Неверный размер для доступа к данным.	Правильно задайте размеры данных для доступа к переменным и параметрам.
16#00001103	Указан адрес вне допустимого диапазона.	Укажите адрес в пределах допустимого диапазона.
16#00001104	Превышен диапазон адресов.	• Укажите адрес в пределах допустимого диапазона. • Исправьте настройки в таблице логических связей.
16#00001106	Указан незарегистрированный номер коммуникационной последовательности.	Исправьте номер коммуникационной последовательности или добавьте последовательность с помощью CX-Protocol.
16#00001109	Произошла ошибка взаимосвязи.	• Устраните несоответствие размеров друг другу в данных команды. • Исправьте настройки в таблице логических связей.
16#0000110A	Данные избыточны.	• Отмените текущий процесс или дождитесь его завершения, прежде чем выполнять команду. • Исправьте настройки в таблице логических связей.
16#0000110B	Слишком длинный ответ.	Правильно задайте количество элементов в массиве команды.
16#0000110C	Другая ошибка параметров.	Правильно настройте массив команды.
16#00002002	Данные защищены.	Снимите защиту и выполните команду еще раз.
16#00002003	Ни одна таблица не зарегистрирована.	Правильно настройте таблицу.
16#00002004	Нет данных, соответствующих искомым данным.	Правильно настройте данные для поиска.
16#00002005	Соответствующего номера программы не существует.	Задайте действительный номер программы.
16#00002006	Соответствующего файла не существует.	Правильно задайте имя файла, включая имена подкаталогов.
16#00002007	Произошла ошибка проверки.	• Проверьте содержимое памяти и запишите правильные данные. • Проверьте содержимое файла.
16#00002101	Доступ невозможен, так как область предназначена только для чтения.	Снимите защиту от записи и выполните команду еще раз.
16#00002102	Данные защищены или запись в таблицу логических связей невозможна.	• Снимите защиту от записи и выполните команду еще раз. • Задайте системные параметры в таблице логических связей.
16#00002103	Регистрация невозможна.	• Создайте файл после удаления ненужных файлов или подготовьте новую файловую память. • Закройте открытые файлы и выполните команду еще раз.
16#00002105	Соответствующего номера программы не существует.	Задайте действительный номер программы.
16#00002106	Соответствующего файла не существует.	Правильно задайте имя файла, включая имена подкаталогов.
16#00002107	Файл с таким именем уже существует.	Измените имя файла для записи и выполните команду еще раз.
16#00002108	Изменение недопустимо, так как оно вызывает ошибку.	Скорректируйте настройки.
16#00002201	Операция невозможна, так как макрос протокола уже выполняется.	Используйте флаг выполнения макроса протокола в качестве нормально замкнутого входа программы.

Значение	Ошибка	Способ устранения
16#00002202	Неверный режим работы.	Проверьте режим работы.
16#00002203	Неверный режим работы для команды (режим «Программирование»).	Проверьте режим работы контроллера.
16#00002204	Неверный режим работы для команды (режим «Отладка»).	Проверьте режим работы контроллера.
16#00002205	Неверный режим работы для команды (режим «Мониторинг»).	Проверьте режим работы контроллера.
16#00002206	Неверный режим работы для команды (режим «Выполнение»).	Проверьте режим работы контроллера.
16#00002207	Указанный узел не является опрашивающим узлом.	Проверьте, какой из узлов сети является опрашивающим узлом.
16#00002208	Неверный режим работы для команды.	Проверьте состояние активации шага.
16#00002211	Модуль занят.	Увеличьте количество повторных попыток или скорректируйте систему, чтобы коммуникационный трафик не концентрировался на соответствующем модуле.
16#00002301	Файлового устройства не существует.	Вставьте носитель. Или отформатируйте память EM.
16#00002302	Файловой памяти нет.	Проверьте устройство файловой памяти.
16#00002303	Встроенных часов нет.	Проверьте технические характеристики модели.
16#00002401	Ошибка контрольной суммы в данных макроса протокола или передача данных еще не завершена.	Передайте данные макроса протокола из CX-Protocol еще раз.
16#00002502	Имеется ошибка в памяти.	Передайте в память правильные данные.
16#00002503	Зарегистрированная конфигурация модулей ввода-вывода не соответствует фактической конфигурации модулей.	Проверьте конфигурацию модулей ввода-вывода.
16#00002504	Слишком много локальных или удаленных точек ввода-вывода.	Правильно задайте количество локальных и удаленных точек ввода-вывода.
16#00002505	Произошла ошибка при обмене данными между модулем ЦПУ и модулем шины ЦПУ.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#00002506	Один и тот же номер стойки, номер модуля или адрес ввода/вывода задан более одного раза.	Исправьте настройки, чтобы каждый номер был уникальным.
16#00002507	Произошла ошибка при обмене данными между модулем ЦПУ и модулем ввода-вывода.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#00002509	Ошибка передачи данных в SYSMAC BUS/2.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#0000250A	Произошла ошибка при передаче данных модуля шины ЦПУ.	Проверьте модули и соединительные кабели. После устранения ошибки выполните команду для сброса ошибки.
16#0000250D	Один и тот же канал используется более одного раза.	Правильно настройте каналы ввода-вывода.

Значение	Ошибка	Способ устранения
16#0000250F	Имеется ошибка в памяти.	• В случае внутренней памяти: запишите правильные данные и выполните команду еще раз. • В случае карты памяти или файловой памяти EM: выполните команду форматирования памяти расширения. • Если указанные выше меры не помогают устранить ошибку, замените память.
16#00002510	Неверно настроена конечная станция.	Правильно задайте конечную станцию.
16#00002601	Защита уже снята.	Снимать защиту не требуется.
16#00002602	Введен неверный пароль.	Укажите правильный пароль.
16#00002604	Данные защищены.	• Снимите защиту от записи и выполните команду еще раз. • Дождитесь завершения службы, которая выполняется в данный момент, или остановите службу и снова выполните команду.
16#00002605	Служба занята.	Дождитесь завершения службы, которая выполняется в данный момент, или остановите службу и снова выполните команду.
16#00002606	Служба остановлена.	При необходимости выполните соответствующую службу.
16#00002607	У вас нет права на выполнение.	• Выполните операцию с узла, с которого был произведен доступ к каналу данных. • Если ошибка все еще присутствует после перезапуска, замените контроллер.
16#00002608	Окружение не настроено.	Выполните необходимую настройку.
16#00002609	Необходимые параметры не заданы.	Задайте необходимые параметры.
16#0000260A	Указанный номер уже определен.	Укажите номер действия или номер перехода, который еще не зарегистрирован, и выполните команду еще раз.
16#0000260B	Невозможно сбросить ошибку.	Устраните причину ошибки, а затем выполните команду сброса ошибки.
16#00003001	У вас нет прав доступа.	Подождите, пока будет разрешен доступ, а затем снова выполните команду.
16#00004001	Служба была прервана.	Устраните причину, по которой была прервана служба, и выполните команду еще раз.

Примечание Помимо кодов, приведенных в таблице выше, о наличии ошибки могут свидетельствовать биты 6, 7 и 15 в коде завершения, которые принимают значение ИСТИНА в случае ошибки. Если значение бита 6 или 7 = ИСТИНА, значит имеется ошибка в модуле ЦПУ в месте назначения. Если значение бита 15 = ИСТИНА, значит произошла ошибка при промежуточной передаче данных по сети.

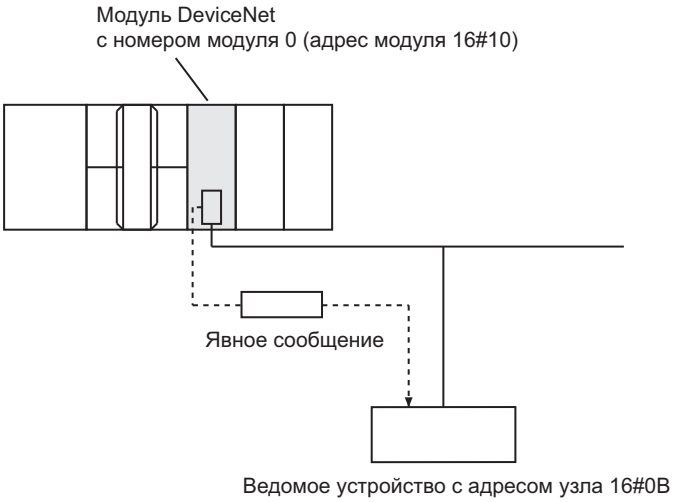
Пример программы

В данном примере команда SendCmd передает явное сообщение через модуль интерфейса DeviceNet. Производится чтение идентификатора производителя из ведомого устройства с адресом узла 16#0B через модуль интерфейса DeviceNet с адресом модуля 16#10.

В таблице ниже указаны используемые параметры связи.

Параметр	Описание
Адрес модуля интерфейса DeviceNet	16#10
Адрес узла ведомого устройства	16#0B
Код службы	16#0E
Идентификатор класса	1

Параметр	Описание
Идентификатор экземпляра	1
Идентификатор атрибута	1
Время ожидания	2,0 с
Количество повторных попыток	2



Массив команды SendDat[] и массив хранения ответа RecvDat[]

В следующих таблицах приведено содержимое массивов SendDat[] (массив команды) и RecvDat[] (массив хранения ответа).

● Массив команды: массив типа BYTE

Элемент массива	Параметр	Содержание	Значение
SendDat[0]	Код команды	Код команды для передачи явного сообщения: 16#2801.	BYTE#16#28
SendDat[1]			BYTE#16#01
SendDat[2]	Адрес узла ведомого устройства	Адрес узла: 16#0B.	BYTE#16#0B
SendDat[3]	Код службы	Код службы для чтения значения указанного атрибута («Запрос одиночного атрибута»): 16#0E.	BYTE#16#0E
SendDat[4]	Идентификатор класса	Идентификатор класса для объекта идентификации (Identity): 16#0001.	BYTE#16#00
SendDat[5]			BYTE#16#01
SendDat[6]	Идентификатор экземпляра	---	BYTE#16#00
SendDat[7]			BYTE#16#01
SendDat[8]	Идентификатор атрибута	Идентификатор атрибута для идентификатора производителя (Vendor ID): 16#01.	BYTE#16#01

● Массив хранения ответа: массив типа BYTE

Элемент массива	Параметр	Содержание
RecvDat[0]	Код команды	Код команды для передачи явного сообщения: 16#2801.
RecvDat[1]		
RecvDat[2]	Код завершения	Код завершения для нормального завершения: 16#0000.
RecvDat[3]		

Элемент массива	Параметр	Содержание
RecvDat[4]	Количество принятых байтов данных после адреса узла ведомого устройства	4 байта
RecvDat[5]		
RecvDat[6]	Адрес узла ведомого устройства	Адрес узла для нормального завершения: 16#0B.
RecvDat[7]	Код службы	Код службы для нормального завершения: 16#8E.
RecvDat[8]	Идентификатор производителя	Идентификатор производителя ведомого устройства
RecvDat[9]		

Определения глобальных переменных

● Глобальные переменные

Имя	Тип данных	Параметр «AT»*1	Комментарий
DeviceNet_OnlineSta	BOOL	IOBus://rack#0/slot#0/Unit2Sta/OnlineSta	В сети

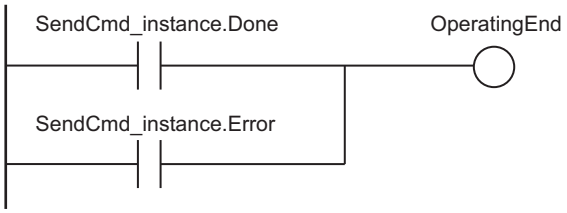
*1. Параметр «AT» для случая, когда модуль последовательного интерфейса установлен в слот с номером 0 стойки с номером 0.

Программа на языке LD

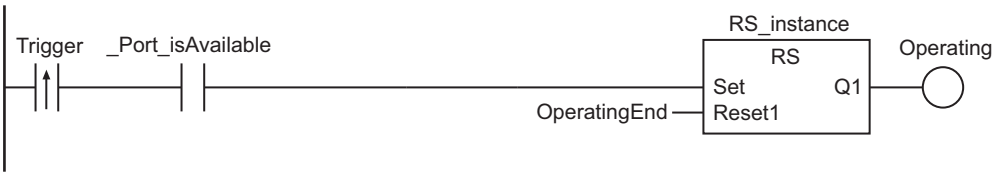
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	InDNetAdr	_sDNET_ADR	(NetNo:=0, NodeNo:=0, UnitNo:=16#0)	Адрес сети назначения
	InOption	_sRESPONSE	(isNonResp:=FALSE, TimeOut:=0, Retry:=0)	Ответ
	SendDat	ARRAY[0..8] OF BYTE	[9(16#0)]	Передаваемые данные
	RecvDat	ARRAY[0..9] OF BYTE	[10(16#0)]	Принимаемые данные
	RS_instance	RS		
	SendCmd_instance	SendCmd		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Port_isAvailable	BOOL	Флаг активации команд сетевой связи

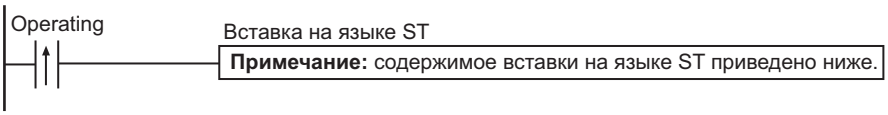
Проверка завершения выполнения команды SendCmd.



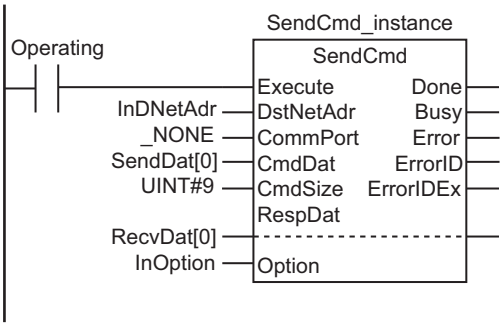
Прием условия выполнения.



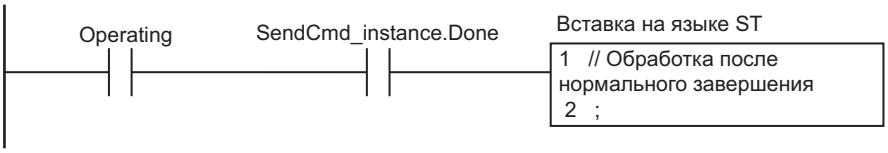
Настройка параметров связи.



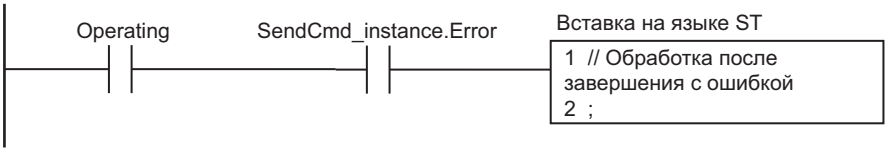
Выполнение команды SendCmd.



Обработка после нормального завершения



Обработка после завершения с ошибкой



● Содержимое вставки на языке ST

```
InDNetAdr.NetNo      :=USINT#0;     // Установка адреса сети.
InDNetAdr.NodeNo     :=USINT#0;
InDNetAdr.UnitNo     :=BYTE#16#10;
InOption.isNonResp   :=FALSE;       // Установка ответа.
InOption.TimeOut     :=UINT#20;
InOption.Retry       :=USINT#2;
SendDat[0]           :=BYTE#16#28;  // Установка массива команды.
SendDat[1]           :=BYTE#16#01;
SendDat[2]           :=BYTE#16#0B;
SendDat[3]           :=BYTE#16#0E;
SendDat[4]           :=BYTE#16#00;
SendDat[5]           :=BYTE#16#01;
SendDat[6]           :=BYTE#16#00;
```

```
SendDat[7]          :=BYTE#16#01;
SendDat[8]          :=BYTE#16#01;
```

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась
	Operating	BOOL	ЛОЖЬ	Обработка
	InDNetAdr	_sDNET_ADR	(NetNo:=0, NodeNo:=0, UnitNo:=16#0)	Адрес сети назначения
	InOption	_sRESPONSE	(isNonResp:=FALSE, TimeOut:=0, Retry:=0)	Ответ
	SendDat	ARRAY[0..8] OF BYTE	[9(16#0)]	Передаваемые данные
	RecvDat	ARRAY[0..9] OF BYTE	[10(16#0)]	Принимаемые данные
	SendCmd_instance	SendCmd		

Внешние переменные	Переменная	Тип данных	Комментарий
	DeviceNet_OnlineSta	BOOL	В сети
	_Port_isAvailable	BOOL	Флаг активации команд сетевой связи

```
// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Port_isAvailable=TRUE)
      AND (DeviceNet_OnlineSta=TRUE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Настройка параметров связи и инициализация команды SendCmd.
IF (OperatingStart=TRUE) THEN
    SendCmd_instance(
        Execute  :=FALSE,
        DstNetAdr:=InDNetAdr,
        CommPort :=_NONE,
        CmdDat   :=SendDat[0],
        CmdSize  :=UINT#9,
        RespDat  :=RecvDat[0],
        Option   :=InOption);
    InDNetAdr.NetNo     :=USINT#0;      //  Установка адреса сети.
    InDNetAdr.NodeNo    :=USINT#0;
    InDNetAdr.UnitNo    :=BYTE#16#10;
```

```
    InOption.isNonResp  :=FALSE;        //  Установка ответа.
    InOption.TimeOut    :=UINT#20;
    InOption.Retry      :=USINT#2;
    SendDat[0]          :=BYTE#16#28;   //  Установка массива команды.
    SendDat[1]          :=BYTE#16#01;
    SendDat[2]          :=BYTE#16#0B;
    SendDat[3]          :=BYTE#16#0E;
    SendDat[4]          :=BYTE#16#00;
    SendDat[5]          :=BYTE#16#01;
    SendDat[6]          :=BYTE#16#00;
    SendDat[7]          :=BYTE#16#01;
    SendDat[8]          :=BYTE#16#01;
    OperatingStart      :=FALSE;
END_IF;
// Выполнение команды SendCmd.
IF (Operating=TRUE) THEN
    SendCmd_instance(
        Execute  :=TRUE,
        DstNetAdr:=InDNetAdr,
        CommPort :=_NONE,
        CmdDat   :=SendDat[0],
        CmdSize  :=UINT#9,
        RespDat  :=RecvDat[0],
        Option   :=InOption);

    IF (SendCmd_instance.Done=TRUE) THEN
        // Обработка после нормального завершения
        Operating:=FALSE;
    END_IF;

    IF (SendCmd_instance.Error=TRUE) THEN
        // Обработка после завершения с ошибкой
        Operating:=FALSE;
    END_IF;
END_IF;
```

NX_SerialSend

Команда NX_SerialSend выполняет передачу данных в беспротокольном режиме через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_SerialSend	Передача данных в беспротокольном режиме	FB	NX_SerialSend_instance NX_SerialSend Execute — Done DevicePort — Busy SendDat — CommandAborted SendSize — Error SendCfg — ErrorID Option Abort	NX_SerialSend_instance(Execute, DevicePort, SendDat, SendSize, SendCfg, Option, Abort, Done, Busy, CommandAborted, Error, ErrorID);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
SendDat[] (массив)	Массив передаваемых данных		Массив передаваемых данных	Зависит от типа данных.	---	*1
SendSize	Объем передаваемых данных		Объем передаваемых данных	0...4096	Байты	0
SendCfg	Условия, прикрепляемые к передаваемым данным		Условия, прикрепляемые к передаваемым данным	---	---	---
Option	Дополнительные параметры		Дополнительные параметры	---	---	---
Abort	Прерывание		Прерывание выполнения команды	Зависит от типа данных.	---	ЛОЖЬ
CommandAborted	Завершение по прерыванию	Выход	Завершение по прерыванию	Зависит от типа данных.	---	---

*1. Если опустить входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1433.																			
SendDat[] (массив)		ОК																		
SendSize							ОК													
SendCfg	Подробные сведения о структуре _sSERIAL_CFG см. в разделе *Функция* на стр. 2-1433.																			
Option	Подробные сведения о структуре _sSERIAL_SEND_OPTION см. в разделе *Функция* на стр. 2-1433.																			
Abort	ОК																			
CommandAborted	ОК																			

Функция

Команда NX_SerialSend выполняет передачу данных в беспротокольном режиме (No-protocol) через указанный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы.

Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

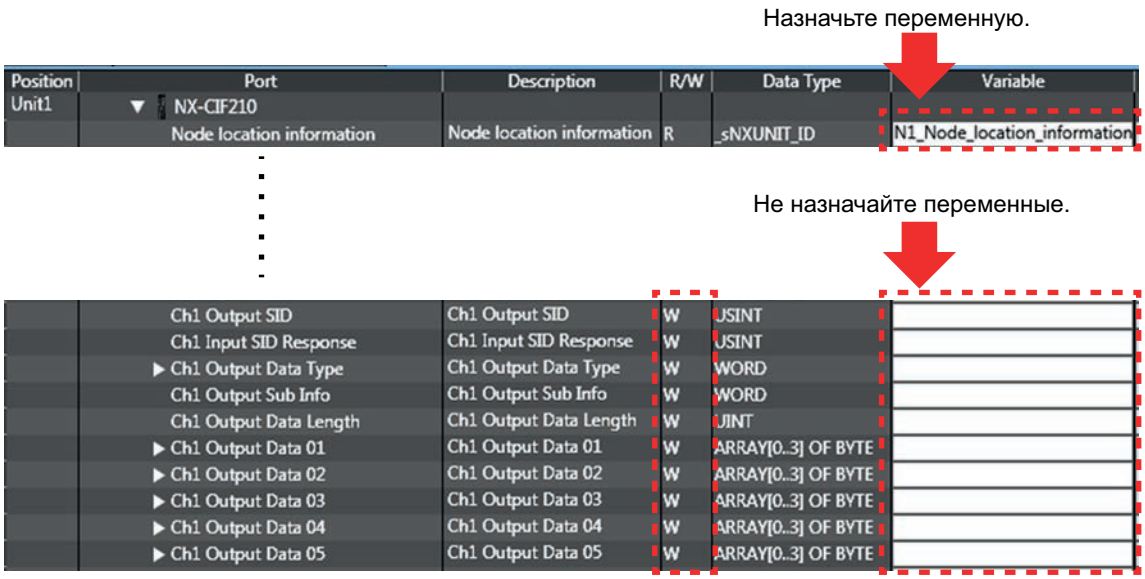
Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.

Задайте значение _DeviceNXUnit для модуля NX или значение _DeviceOptionBoard для дополнительной платы.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

Чтобы указать модуль NX, используйте переменную *NxUnit* для указания устройства.
Переменные *EcatSlave* и *OptBoard* в этом случае не используются.
В переменную *NxUnit* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.
Чтобы указать дополнительную плату, используйте переменную *OptBoard* для указания устройства.
Переменные *NxUnit* и *EcatSlave* в этом случае не используются.
В переменную *OptBoard* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода, следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2
Для модуля NX укажите порт 1 или порт 2.
Для дополнительной платы укажите порт 1.

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.

Перечислитель	Значение
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceNXUnit или _DeviceOptionBoard.

Передаваемые данные указываются с помощью входной переменной *SendDat*. Объем передаваемых данных указывается с помощью входной переменной *SendSize*.
Если значение *SendSize = 0*, то ничего не передается. При выполнении команды вместо переменной *Busy* в состояние ИСТИНА перейдет переменная *Done*.

Если к передаваемым данным требуется присоединить код начала и код конца данных, эти коды следует задать во входной переменной *SendCfg*.
Для входной переменной *SendCfg* используется структурный тип данных _sSERIAL_CFG. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
SendCfg	Условия, прикрепляемые к передаваемым данным	Условия, прикрепляемые к передаваемым данным	_sSERIAL_CFG	---	---	---
StartTrig	Наличие кода начала данных	Наличие кода начала данных	_eSERIAL_START	_SERIAL_START_NONE _SERIAL_START_STARTCODE1 _SERIAL_START_STARTCODE2	---	_SERIAL_START_NONE
StartCode	Код начала данных	Код начала данных	BYTE[2]	Зависит от типа данных.	---	[2(16#0)]
EndTrig	Наличие кода конца данных	Наличие кода конца данных	_eSERIAL_END	_SERIAL_END_NONE _SERIAL_END_ENDCODE1 _SERIAL_END_ENDCODE2 _SERIAL_END_TERMINATION_CHAR _SERIAL_END_RCV_SIZE	---	_SERIAL_END_NONE
EndCode	Код конца данных	Код конца данных	BYTE[2]	Зависит от типа данных.	---	[2(16#0)]
RcvSizeCfg	Объем принятых данных	В этой команде не используется.	UINT	0...4096	Байты	0

Для переменной *StartTrig* используется перечислимый тип данных _eSERIAL_START.
Значения перечислителей перечислимого типа _eSERIAL_START приведены в таблице ниже.

Перечислитель	Значение
_SERIAL_START_NONE	Нет
_SERIAL_START_STARTCODE1	1-байтовый код
_SERIAL_START_STARTCODE2	2-байтовый код

Для переменной *EndTrig* используется перечислимый тип данных _eSERIAL_END.
Значения перечислителей перечислимого типа _eSERIAL_END приведены в таблице ниже.

Перечислитель	Значение
_SERIAL_END_NONE	Нет
_SERIAL_END_ENDCODE1	1-байтовый код

Перечислитель	Значение
_SERIAL_END_ENDCODE2	2-байтовый код
_SERIAL_END_TERMINATION_CHAR	Условие завершения
_SERIAL_END_RCV_SIZE	Объем принятых данных

Подробную информацию о действии кодов начала и конца данных см. в разделе *Действие кодов начала и конца данных* на стр. 2-1449.

Чтобы данные передавались из контроллера в интерфейсный модуль серии NX с некоторой задержкой, задайте время задержки (в приращениях 0,01 с) с помощью входной переменной *Option.SendDelay*.
Для входной переменной *Option* используется структурный тип данных _sSERIAL_SEND_OPTION. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Option	Дополнительные параметры	Дополнительные параметры	_sSERIAL_SEND_OPTION	---	---	---
SendDelay	Время задержки передачи	Время задержки передачи	UINT	Зависит от типа данных.	0,01 с	0

Меры предосторожности для обеспечения надлежащей эксплуатации

Если эта команда будет выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля, произойдет ошибка.

Действие кодов начала и конца данных

Используйте переменные *SendCfg.StartTrig* и *SendCfg.EndTrig* для указания условий использования кодов начала и конца данных, которые прикрепляются к передаваемым данным.
Длина кода начала или кода конца данных, присоединяемого к передаваемым данным, не должна включаться в значение, задаваемое во входной переменной *SendSize*.
Действие переменных *StartTrig* и *EndTrig* поясняется ниже.

Значение *StartTrig*	Действие
_SERIAL_START_NONE	---
_SERIAL_START_STARTCODE1	Данные *SendDat* передаются с кодом начала данных, который присоединяется к началу данных. Пример: STX
_SERIAL_START_STARTCODE2	

Значение *EndTrig*	Действие
_SERIAL_END_NONE	---
_SERIAL_END_ENDCODE1	Данные *SendDat* передаются с кодом конца данных, который присоединяется к концу данных. Пример: ETX
_SERIAL_END_ENDCODE2	
_SERIAL_END_TERMINATION_CHAR	Ошибка
_SERIAL_END_RCV_SIZE	Ошибка

Прерывание выполнения команды

Если вход *Abort* переходит в состояние ИСТИНА во время выполнения команды, выполнение команды прерывается.
Когда выполнение команды прерывается, значение переменной *CommandAborted* меняется на ИСТИНА. Выполнение команды прерывается, даже если в этот момент передаются данные.
Если выполнение команды уже завершилось к тому моменту, когда делается попытка прервать ее выполнение, значение *Done* меняется на ИСТИНА и команда завершается нормально.
Если входы *Abort* и *Execute* переходят в состояние ИСТИНА одновременно, значение *CommandAborted* меняется на ИСТИНА.

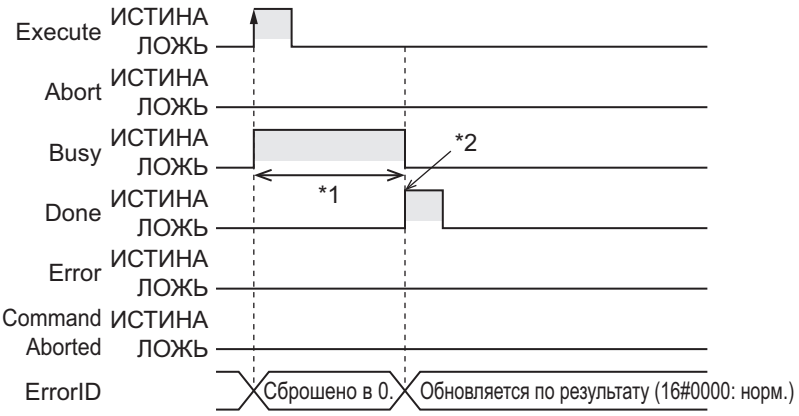
Операция прерывания лишь завершает обработку выхода *Busy*, но она не очищает буфер передачи. Для очистки буфера следует использовать команду *NX_SerialBufClear* на стр. 2-1516.

Временные диаграммы

Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение (когда *SendDelay = 0* (0 с))

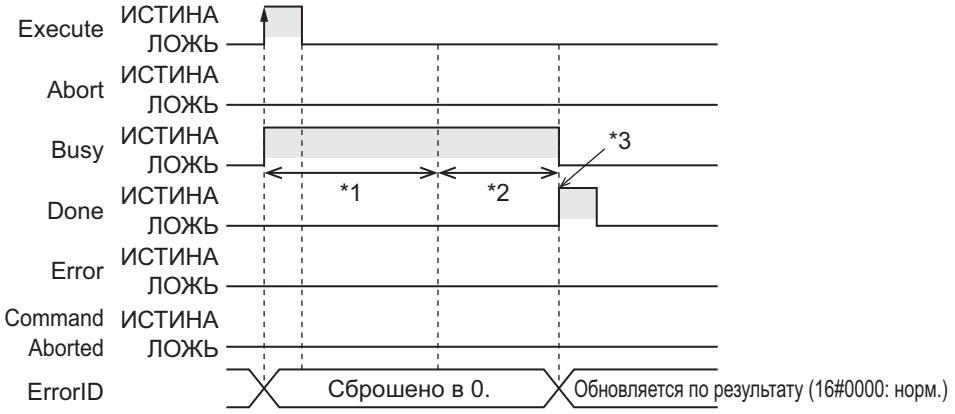
Ниже показана работа команды для случая, когда *SendDelay = 0* (0 с).



*1. Выполняется передача
*2. Передача завершена

● Нормальное завершение (когда *SendDelay = 100* (1 с))

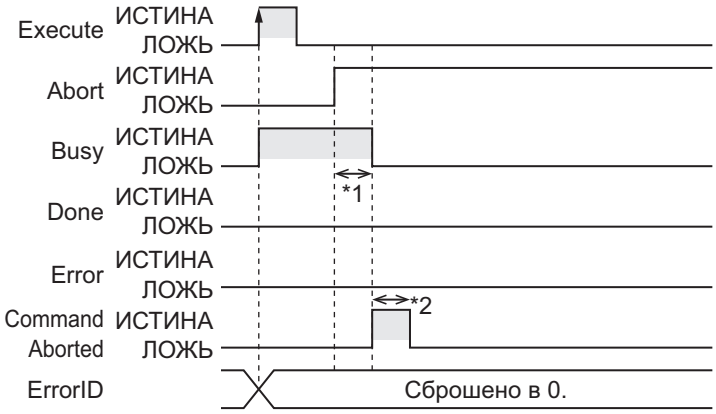
Ниже показана работа команды для случая, когда *SendDelay = 100* (1 с).



*1. Время задержки передачи = 1 с.
*2. Выполняется передача
*3. Передача завершена

● Выполнено прерывание (когда *Busy* = ИСТИНА)

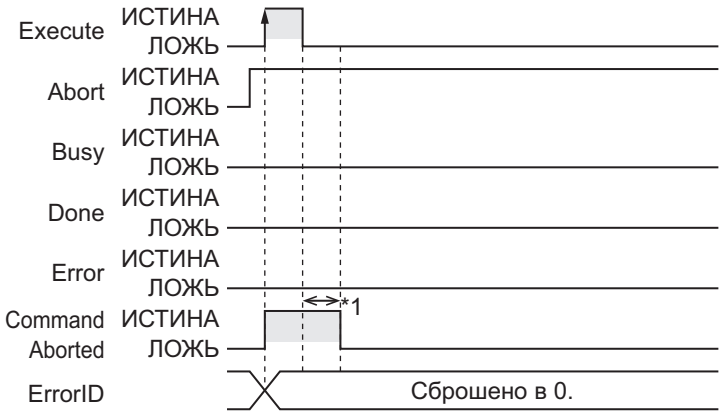
Ниже показана работа команды для случая, когда значение *Abort* меняется на ИСТИНА, когда *Busy* = ИСТИНА.



*1. Обрабатывается прерывание
*2. Меняется на ЛОЖЬ после одного периода выполнения задачи.

● Выполнено прерывание (когда *Execute* = ИСТИНА)

Ниже показана работа команды для случая, когда значения *Abort* и *Execute* меняются на ИСТИНА.



*1. Меняется на ЛОЖЬ после одного периода выполнения задачи.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_OptBoardSta*1	Состояние дополнительной платы	ARRAY[1..2] of _sOPTBOARD_STA	Хранит данные о состоянии дополнительной платы.
_NXB_UnitIOActiveTbl*2	Состояние активности данных ввода-вывода модуля NX	ARRAY[0..32] OF BOOL*3	• Эти данные о состоянии сообщают, может ли тот или иной модуль NX участвовать в обмене данными ввода-вывода. • Индекс элемента массива соответствует номеру модуля NX. Индексу 0 соответствует ведущее устройство шины NX.

*1. Эту переменную можно использовать только с модулями ЦПУ NX1P2.
*2. Эту переменную можно использовать только с модулями ЦПУ NX102 и NX1P2.
*3. Тип данных для модулей ЦПУ NX1P2: *ARRAY [0..8] OF BOOL*.

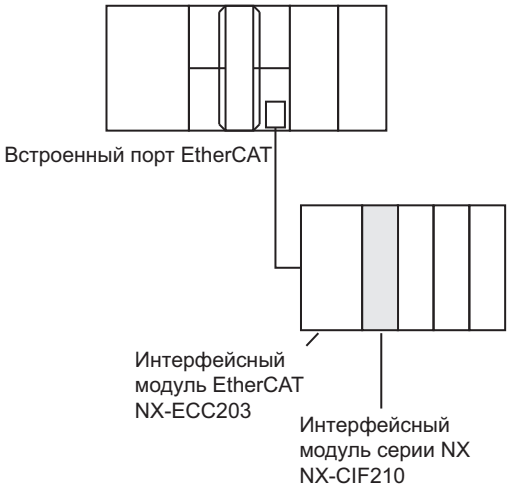
Меры предосторожности для обеспечения надлежащей эксплуатации

- Пока *Abort* будет оставаться в состоянии ЛОЖЬ, выполнение этой команды будет продолжаться до завершения, даже если *Execute* перейдет в состояние ЛОЖЬ или время выполнения превысит период выполнения задачи.
  По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки. Если *Abort* перейдет в состояние ИСТИНА во время выполнения команды, значение *CommandAborted* или *Done* поменяется на ИСТИНА.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для переменной *SendSize*, *SendCfg.StartTrig*, *SendCfg.EndTrig*, *DevicePort.DevicePortType* или *DevicePort.PortNo* задано значение вне допустимого диапазона.

b) Размер переменной-массива, указанной для *SendDat*, меньше размера, указанного в *SendSize*.
c) Модуля, дополнительной платы или порта, указанного с помощью *DevicePort*, не существует.
d) Недопустимый тип данных переменной *DevicePort*.
e) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
f) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.
Этой другой командой может быть одна из следующих команд: команда NX_SerialSend, команда NX_ModbusRtuCmd, команда NX_ModbusRtuRead и команда NX_ModbusWrite.
g) Эта команда была выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля.
h) В качестве режима связи по последовательному интерфейсу для указанной дополнительной платы не установлен режим *Без протокола (No-protocol)*.

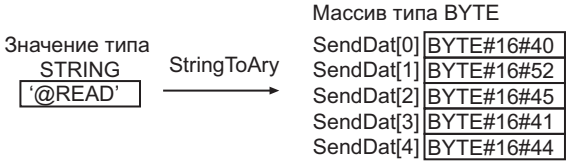
Пример программы

В данном примере интерфейсный модуль серии NX (NX-CIF210) подключен к интерфейсному модулю EtherCAT (NX-ECC203).
Значение номера модуля, установленное для модуля NX-CIF210: *1*.



Передается команда связи в беспротокольном режиме считывателю штрих-кодов, который подключен к последовательному порту 2 модуля NX-CIF210. Передается команда чтения номера режима съемки (@READ).

При передаче команды используется команда StringToAry, которая разбивает текстовую строку «@READ» на отдельные символы и преобразует их в коды символов. Коды символов сохраняются в элементы массива SendDat[].



Код начала данных не используется. Используется код конца данных: 16#OD (CR).

В следующей таблице приведены значения параметров модуля NX-CIF210.

Параметр	Заданное значение
Порт 2: скорость передачи данных	38 400 бит/с
Порт 2: длина данных	8 бит
Порт 2: проверка четности	Нет
Порт 2: количество стоп-битов	1 бит
Порт 2: управлением потоком данных	Нет

Определения глобальных переменных

● Глобальные переменные

Имя	Тип данных	AT	Комментарий
E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	ECAT://node#1/NX Unit I/O Data Active Status 125	Использование данных ввода-вывода для 63 модулей NX.
N1_Node_location_information	_sNXUNIT_ID	---	Переменная устройства для указания модуля NX-CIF210*1

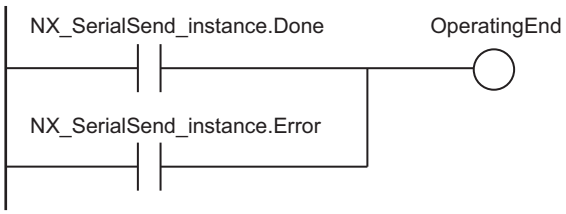
*1. В Sysmac Studio щелкните правой кнопкой мыши ведомый терминал серии NX, выберите **Display Node Location Port (Отобразить порт расположения узла)** и задайте переменную устройства. Дополнительные сведения см. в: *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Программа на языке LD

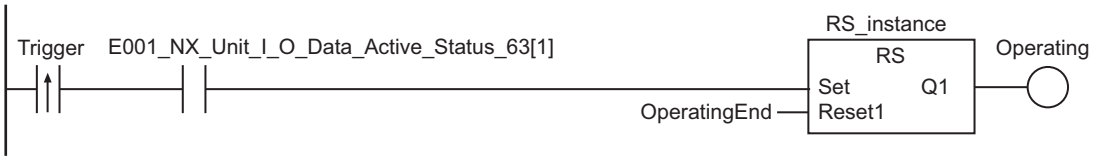
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperationEnd	BOOL	ЛОЖЬ	Обработка завершена
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	DevicePort	_sDEVICE_PORT		Параметры порта
	SendDat	ARRAY [0..5] OF BYTE	[6(16#0)]	Передаваемые данные
	SendSize	UINT	0	Объем передаваемых данных
	RS_instance	RS		
	NX_SerialSend_instance	NX_SerialSend		
	SendCfg	_sSERIAL_SEND_CFG		
	StartTrig	_eSERIAL_START	_SERIAL_START_NONE	Не использовать код начала
	StartCode	BYTE[2]	[2(16#0)]	
	EndTrig	_eSERIAL_END	_SERIAL_END_ENDCODE1	Использовать код конца
	EndCode	BYTE[2]	[16#0D,16#00]	16#0D(CR)

Внешние переменные	Переменная	Тип данных	Комментарий
	E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	• Использование данных ввода-вывода для 63 модулей NX. • Если номер соответствующего модуля = *1*, используется E001_NX_Unit_I_O_Data_Active_Status_63[1].
	N1_Node_location_information	_sNXUNIT_ID	Переменная устройства для указания модуля NX-CIF210

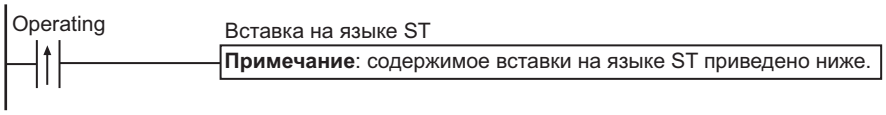
Определение, завершилось ли выполнение команды NX_SerialSend.



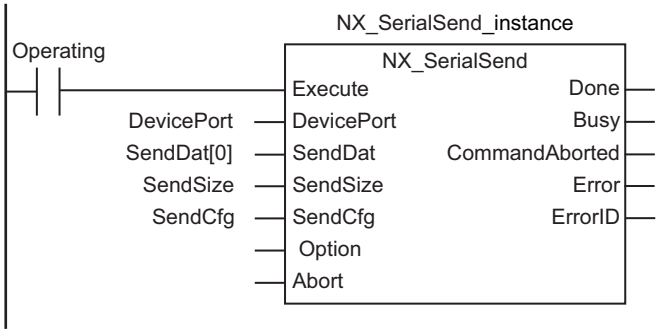
Прием условия выполнения.



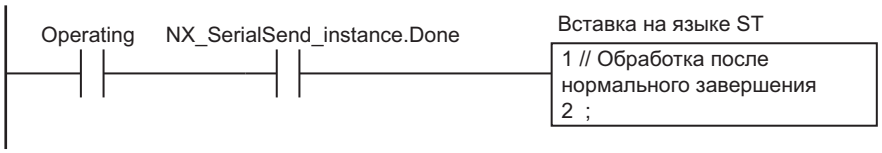
Настройка параметров связи.

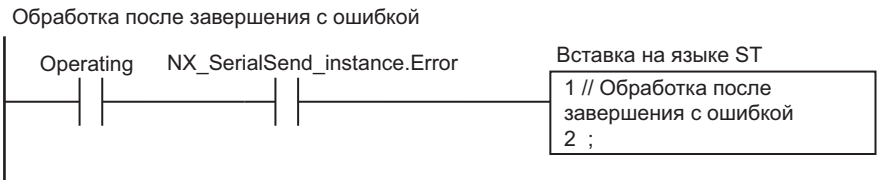


Выполнение команды NX_SerialSend.



Обработка после нормального завершения





● Содержимое вставки на языке ST

```
DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
DevicePort.NxUnit:=N1_Node_location_information;
DevicePort.PortNo:=2;
StringToAry(In:='@READ', AryOut:=SendDat[0]);
SendSize := UINT#10#5;
```

Программа на языке ST

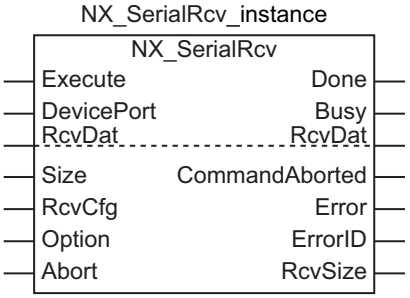
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась
	Operating	BOOL	ЛОЖЬ	Обработка
	DevicePort	_sDEVICE_PORT		Параметры порта
	SendDat	ARRAY[0..5] OF BYTE	[6(16#0)]	Передаваемые данные
	SendSize	UINT	0	Объем передаваемых данных
	NX_SerialSend_instance	NX_SerialSend		
	SendCfg	_sSERIAL_CFG		
	StartTrig	_eSERIAL_START	_SERIAL_START_NONE	Не использовать код начала
	StartCode	BYTE[2]	[2(16#0)]	
	EndTrig	_eSERIAL_END	_SERIAL_END_ENDCODE1	Использовать код конца
	EndCode	BYTE[2]	[16#0D, 16#00]	16#0D(CR)

Внешние переменные	Переменная	Тип данных	Комментарий
	E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	• Использование данных ввода-вывода для 63 модулей NX. • Если номер соответствующего модуля = *1*, используется E001_NX_Unit_I_O_Data_Active_Status_63[1].
	N1_Node_location_information	_sNXUNIT_ID	Переменная устройства для указания модуля NX-CIF210

```
// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE)
    AND (E001_NX_Unit_I_O_Data_Active_Status_63[1]) AND (NX_SerialSend_instance.Bus
y=FALSE) ) THEN
        OperatingStart:=TRUE;
        Operating:=TRUE;
        DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
        DevicePort.NxUnit:=N1_Node_location_information;
        DevicePort.PortNo:=2;
END_IF;
LastTrigger:=Trigger;

// Настройка параметров связи и инициализация команды NX_SerialSend.
IF (OperatingStart=TRUE) THEN
         NX_SerialSend_instance(
                          Execute:=FALSE,
                          DevicePort:=DevicePort;
                          SendDat:=SendDat[0],
                          SendSize:=UINT#1,
                          SendCfg:=SendCfg);
         StringToAry(In:='@READ', AryOut:=SendDat[0]);
         SendSize:=UINT#10#5;
         OperatingStart:=FALSE;
END_IF;

// Выполнение команды NX_SerialSend.
IF (Operating=TRUE) THEN
         NX_SerialSend_instance(
                          Execute:=TRUE,
                          DevicePort:=DevicePort, // Параметры порта
                          SendDat:=SendDat[0],    // Передаваемые данные
                          SendSize:=SendSize,     // Объем передаваемых данных
                          SendCfg:=SendCfg);      // Настройки кода конца данных
         IF (NX_SerialSend_instance.Done=TRUE) THEN
                  // Обработка после нормального завершения

                  Operating:=FALSE;
         END_IF;

         IF (NX_SerialSend_instance.Error=TRUE) THEN
                 // Обработка после завершения с ошибкой

                 Operating:=FALSE;
         END_IF;
END_IF;
```

NX_SerialRcv

Команда NX_SerialRcv выполняет чтение данных в беспротокольном режиме через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_SerialRcv	Прием данных в беспротокольном режиме	FB		NX_SerialRcv_instance(Execute, DevicePort, RcvDat, Size, RcvCfg, Option, Abort, Done, Busy, CommandAborted, Error, ErrorID, RcvSize);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
Size	Объем сохраняемых данных		Объем *RcvDat* в байтах	1...4096	Байты	1
RcvCfg	Настройка завершения приема		Настройка завершения приема	---	---	---
Option	Дополнительные параметры		Дополнительные параметры	---	---	---
Abort	Прерывание		Прерывание выполнения команды	---	---	ЛОЖЬ
RcvDat[] (массив)	Принимаемые данные	Вход-выход	Переменная для хранения данных, полученных из буфера приема	Зависит от типа данных.	---	---
CommandAborted	Завершение по прерыванию	Выход	Завершение по прерыванию	Зависит от типа данных.	---	---
RcvSize	Объем принятых данных		Объем данных, фактически полученных из буфера приема	0...4096	Байты	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1446.																			
Size							ОК													
RcvCfg	Подробные сведения о структуре _sSERIAL_CFG см. в разделе *Функция* на стр. 2-1446.																			
Option	Подробные сведения о структуре _sSERIAL_RCV_OPTION см. в разделе *Функция* на стр. 2-1446.																			
Abort	ОК																			
RcvDat[] (массив)		ОК																		
CommandAborted	ОК																			
RcvSize							ОК													

Функция

Команда NX_SerialRcv выполняет чтение данных в беспротокольном режиме (No-protocol) через указанный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы.

Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

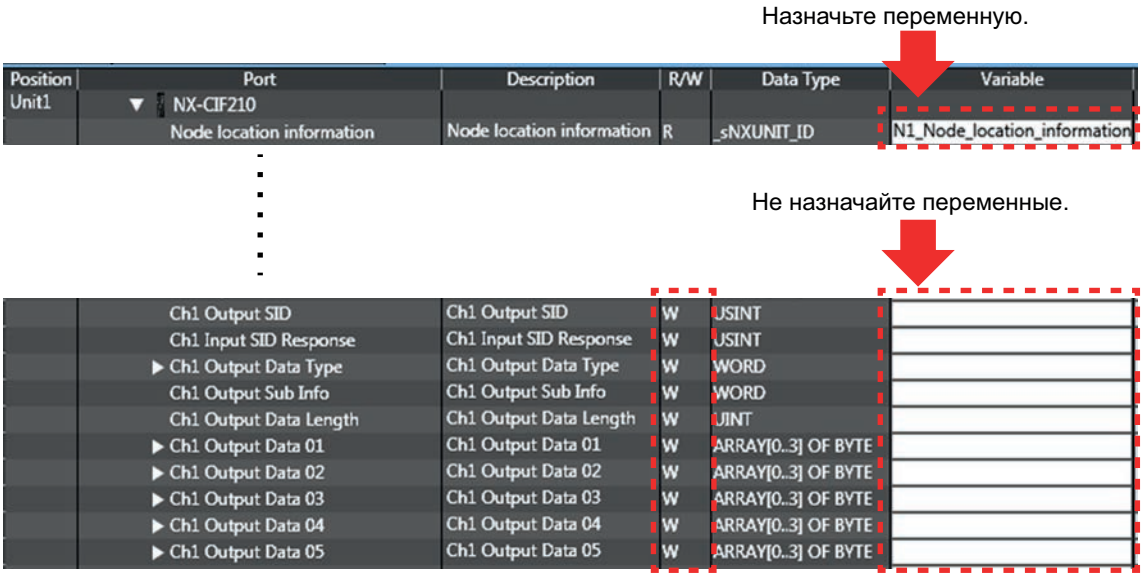
Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.

Задайте значение _DeviceNXUnit для модуля NX или значение _DeviceOptionBoard для дополнительной платы.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

Чтобы указать модуль NX, используйте переменную *NxUnit* для указания устройства.
Переменные *EcatSlave* и *OptBoard* в этом случае не используются.
В переменную *NxUnit* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.
Чтобы указать дополнительную плату, используйте переменную *OptBoard* для указания устройства.
Переменные *NxUnit* и *EcatSlave* в этом случае не используются.
В переменную *OptBoard* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода, следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2
Для модуля NX укажите порт 1 или порт 2.
Для дополнительной платы укажите порт 1.

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.

Перечислитель	Значение
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceNXUnit или _DeviceOptionBoard.

Сначала данные, принятые модулем, сохраняются в буфер приема.
Переменная, в которую должны сохраняться данные, получаемые из буфера приема, указывается с помощью входной-выходной переменной *RcvDat*.
Размер переменной *RcvDat* в байтах указывается с помощью входной переменной *Size*.
В выходную переменную *RcvSize* записывается объем данных, фактически полученных из буфера приема.

Если принимаемые данные включают код начала или код конца данных, необходимо настроить входную переменную *RcvCfg*.
Для входной переменной *RcvCfg* используется структурный тип данных _sSERIAL_CFG. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
RcvCfg	Настройка завершения приема	Настройка завершения приема	_sSERIAL_CFG	---	---	---
StartTrig	Наличие кода начала данных	Наличие кода начала данных	_eSERIAL_START	_SERIAL_START_NONE _SERIAL_START_STARTCODE1 _SERIAL_START_STARTCODE2	---	_SERIAL_START_NONE
StartCode	Код начала данных	Код начала данных	BYTE[2]	Зависит от типа данных.	---	[2(16#0)]
EndTrig	Наличие кода конца данных	Наличие кода конца данных	_eSERIAL_END	_SERIAL_END_NONE _SERIAL_END_ENDCODE1 _SERIAL_END_ENDCODE2 _SERIAL_END_TERMINATION_CHAR _SERIAL_END_RCV_SIZE	---	_SERIAL_END_NONE
EndCode	Код конца данных	Код конца данных	BYTE[2]	Зависит от типа данных.	---	[2(16#0)]
RcvSizeCfg	Объем принятых данных	Объем принимаемых данных, указываемый, когда EndTrig = _SERIAL_END_RCV_SIZE	UINT	0...4 096	Байты	0

Для переменной *StartTrig* используется перечислимый тип данных _eSERIAL_START.
Значения перечислителей перечислимого типа _eSERIAL_START приведены в таблице ниже.

Перечислитель	Значение
_SERIAL_START_NONE	Нет
_SERIAL_START_STARTCODE1	1-байтовый код
_SERIAL_START_STARTCODE2	2-байтовый код

Для переменной *EndTrig* используется перечислимый тип данных _eSERIAL_END.
Значения перечислителей перечислимого типа _eSERIAL_END приведены в таблице ниже.

Перечислитель	Значение
_SERIAL_END_NONE	Нет
_SERIAL_END_ENDCODE1	1-байтовый код
_SERIAL_END_ENDCODE2	2-байтовый код
_SERIAL_END_TERMINATION_CHAR	Условие завершения
_SERIAL_END_RCV_SIZE	Объем принятых данных

Подробную информацию о действии кодов начала и конца данных см. в разделе *Действие кодов начала и конца данных* на стр. 2-1449.

Для настройки дополнительных параметров используется входная переменная *Option*.
Для входной переменной *Option* используется структурный тип данных _eSERIAL_RCV_OPTION. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Option	Дополнительные параметры	Дополнительные параметры	_sSERIAL_RCV_OPTION	---	---	---
TimeOut*1	Время ожидания	Время ожидания	UINT	Зависит от типа данных.	0,1 с	20
LastDatRcv (резерв)	Последний прием данных	Последний прием данных	BOOL	ЛОЖЬ*2	---	ЛОЖЬ
ClearBuf (резерв)	Условие очистки буфера приема	Условие очистки буфера приема	BOOL	Зависит от типа данных.*3	---	ЛОЖЬ

*1. Если обработка не завершается нормально в течение указанного времени, возникает ошибка.
Если *TimeOut = 0*, время ожидания завершения обработки не ограничивается.
*2. Всегда задавайте значение ЛОЖЬ.
*3. Очистка буфера приема не выполняется, что бы ни было указано: ИСТИНА или ЛОЖЬ.

Меры предосторожности для обеспечения надлежащей эксплуатации

Если эта команда будет выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля, произойдет ошибка.

Действие кодов начала и конца данных

Для настройки условия использования кода начала данных для приема данных используется входная переменная *RcvCfg.StartTrig*. А для настройки условия использования кода конца данных для приема данных используется входная переменная *RcvCfg.EndTrig*.
В следующей таблице показано, как работает команда в зависимости от настройки *StartTrig* и *EndTrig*.

StartTrig	EndTrig	Работа
_SERIAL_START_NONE	_SERIAL_END_NONE	Принимаются данные, находящиеся в буфере приема. Если в буфере приема нет принятых данных, в выходную переменную *RcvSize* выдается 0 (байт) и команда приема завершается нормально. Если настроено это условие, считываются данные, оставшиеся в буфере приема, в объеме, который указан во входной переменной Size (объем сохраняемых данных).
	_SERIAL_END_ENDCODE1 _SERIAL_END_ENDCODE2	Из буфера приема принимаются данные в следующем диапазоне: от начала буфера до кода конца данных. Пример: ETX
	_SERIAL_END_TERMINATION_CHAR	Из буфера приема принимаются данные в следующем диапазоне: от начала буфера до данных, распознанных в качестве признака конца данных. *1
	_SERIAL_END_RCV_SIZE	Из буфера приема принимаются данные в следующем диапазоне: от начала буфера до достижения объема принятых данных, указанного в *RcvSizeCfg*. Обработка выполняется только после того, как в буфере накапливается указанное количество данных.
_SERIAL_START_STARTCODE1 _SERIAL_START_STARTCODE2	_SERIAL_END_NONE	Из буфера приема принимаются данные в следующем диапазоне: от кода начала данных до конца данных.
	_SERIAL_END_ENDCODE1 _SERIAL_END_ENDCODE2	Из буфера приема принимаются данные в следующем диапазоне: от кода начала данных до кода конца данных. Пример: ETX
	_SERIAL_END_TERMINATION_CHAR	Из буфера приема принимаются данные в следующем диапазоне: от кода начала данных до данных, распознанных в качестве признака конца данных. *1
	_SERIAL_END_RCV_SIZE	Из буфера приема принимаются данные в следующем диапазоне: от кода начала данных до достижения объема принятых данных, указанного в *RcvSizeCfg*. Обработка выполняется только после того, как в буфере накапливается указанное количество данных.

*1. Если количество символов, распознаваемых в качестве признака конца данных в интерфейсном модуле, установлено равным *0 (Не обнаруживать конец данных)*, прием данных будет завершен, когда будут приняты данные в объеме, который указан во входной переменной *Size* (объем сохраняемых данных).

Меры предосторожности для обеспечения надлежащей эксплуатации

Если условие **_SERIAL_END_TERMINATION_CHAR** будет выбрано, когда указана дополнительная плата, произойдет ошибка.

Работа команды при недостаточном значении объема сохраняемых данных

Если значение входной переменной *Size* (объем сохраняемых принятых данных) меньше, чем фактический объем принятых данных, команда работает в соответствии с настройкой кодов конца и начала данных, как показано в таблице ниже.

StartTrig	EndTrig	Работа
_SERIAL_START_NONE	_SERIAL_END_NONE	Нормальное завершение
	_SERIAL_END_ENDCODE1	Завершается с ошибкой, но данные принимаются. Пример: ETX
	_SERIAL_END_ENDCODE2	
	_SERIAL_END_TERMINATION_CHAR	Завершается с ошибкой, но данные принимаются.*1
	_SERIAL_END_RCV_SIZE	• Завершается с ошибкой проверки входного значения. • Прием данных невозможен.
_SERIAL_START_STARTCODE1 _SERIAL_START_STARTCODE2	_SERIAL_END_NONE	Завершается с ошибкой, но данные принимаются.
	_SERIAL_END_ENDCODE1	Завершается с ошибкой, но данные принимаются. Пример: ETX
	_SERIAL_END_ENDCODE2	
	_SERIAL_END_TERMINATION_CHAR	Завершается с ошибкой, но данные принимаются.*1
	_SERIAL_END_RCV_SIZE	• Завершается с ошибкой проверки входного значения. • Прием данных невозможен.

*1. Если будет указана дополнительная плата, произойдет ошибка.

В переменную *RcvDat* принимаются данные в заданном объеме (объем сохраняемых данных), а остальные данные остаются в буфере приема.
Остающиеся данные могут быть получены при следующем выполнении команды SerialRcv.

Например, если в буфере приема имеется 10 байтов данных, а емкость массива хранения данных *RcvDat* составляет 5 байтов, будет получено 5 байтов данных, а остальные 5 байтов данных останутся в буфере приема. Значение выходной переменной *RcvSize* будет равно 5 (байтов), отражая объем сохраненных данных.

Буфер приема		Массив для хранения принятых данных RcvDat[ ]
1-й байт	Получение принятых данных.	1
2-й байт		2
3-й байт		3
4-й байт		4
5-й байт		5

Буфер приема		Массив для хранения принятых данных RcvDat[ ]
6-й байт	Не может быть сохранен в *RcvDat*. Данные остаются в буфере приема. Эти принятые данные будут получены при выполнении следующей команды NX_SerialRcv.	---
7-й байт		
8-й байт		
9-й байт		
10-й байт		

Прерывание выполнения команды

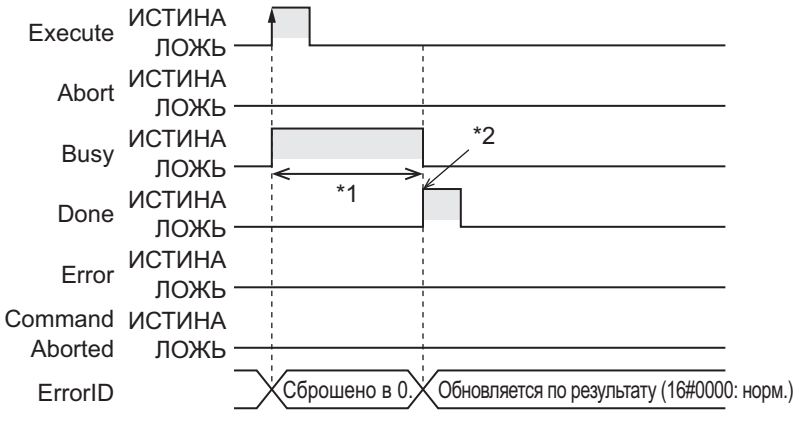
Если вход *Abort* переходит в состояние ИСТИНА во время выполнения команды, выполнение команды прерывается.
Когда выполнение команды прерывается, значение переменной *CommandAborted* меняется на ИСТИНА. Выполнение команды прерывается, даже если в этот момент передаются данные.
Если выполнение команды уже завершилось к тому моменту, когда делается попытка прервать ее выполнение, значение *Done* меняется на ИСТИНА и команда завершается нормально.
Если входы *Abort* и *Execute* переходят в состояние ИСТИНА одновременно, значение *CommandAborted* меняется на ИСТИНА.

Операция прерывания лишь завершает обработку выхода *Busy*, но она не очищает буфер передачи. Для очистки буфера следует использовать команду *NX_SerialBufClear* на стр. 2-1516.

Временные диаграммы

Временные диаграммы работы команды представлены на рисунках ниже.

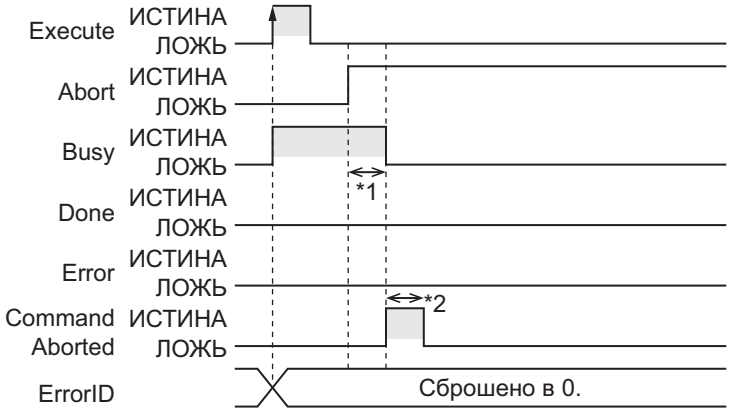
● Нормальное завершение



*1. Обрабатывается прием
*2. Приняты данные в беспротокольном режиме.

● Выполнено прерывание (когда *Busy* = ИСТИНА)

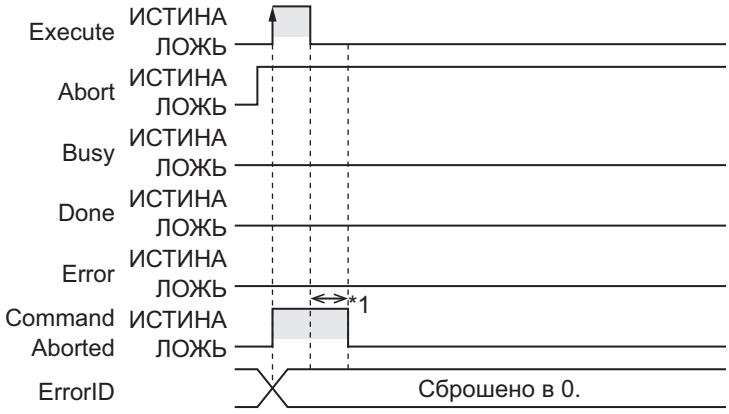
Ниже показана работа команды для случая, когда значение *Abort* меняется на ИСТИНА, когда *Busy* = ИСТИНА.



*1. Обрабатывается прерывание
*2. Меняется на ЛОЖЬ после одного периода выполнения задачи.

● Выполнено прерывание (когда *Execute* = ИСТИНА)

Ниже показана работа команды для случая, когда значения *Abort* и *Execute* меняются на ИСТИНА.



*1. Меняется на ЛОЖЬ после одного периода выполнения задачи.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_OptBoardSta*1	Состояние дополнительной платы	ARRAY[1..2] of _sOPTBOARD_STA	Хранит данные о состоянии дополнительной платы.
_NXB_UnitIOActiveTbl*2	Состояние активности данных ввода-вывода модуля NX	ARRAY[0..32] OF BOOL*3	• Эти данные о состоянии сообщают, может ли тот или иной модуль NX участвовать в обмене данными ввода-вывода. • Индекс элемента массива соответствует номеру модуля NX. Индексу 0 соответствует ведущее устройство шины NX.

*1. Эту переменную можно использовать только с модулями ЦПУ NX1P2.
*2. Эту переменную можно использовать только с модулями ЦПУ NX102 и NX1P2.
*3. Тип данных для модулей ЦПУ NX1P2: *ARRAY [0..8] OF BOOL*.

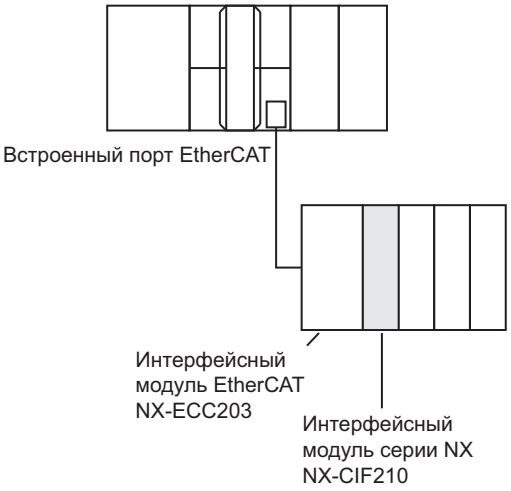
Меры предосторожности для обеспечения надлежащей эксплуатации

- Пока *Abort* будет оставаться в состоянии ЛОЖЬ, выполнение этой команды будет продолжаться до завершения, даже если *Execute* перейдет в состояние ЛОЖЬ или время выполнения превысит период выполнения задачи.
  По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки. Если *Abort* перейдет в состояние ИСТИНА во время выполнения команды, значение *CommandAborted* или *Done* поменяется на ИСТИНА.
- Если *RcvCfg.EndTrig* = _SERIAL_END_RCV_SIZE, а значение входной переменной *RcvCfg.RcvSizeCfg = 0*, данные приняты не будут. В этом случае значение переменной *Done* меняется на ИСТИНА при выполнении команды.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для переменной *RcvCfg.RcvSizeCfg* задано значение вне допустимого диапазона, когда для переменной *RcvCfg.EndTrig* задано значение _SERIAL_END_RCV_SIZE.
  b) Для переменной *Size*, *DevicePort.DevicePortType* или *DevicePort.PortNo* задано значение вне допустимого диапазона.
  c) *Option.LastDatRcv* = ИСТИНА.
  d) Размер переменной-массива, указанной для входной-выходной переменной *RcvDat*, меньше размера, указанного с помощью входной переменной *Size*.
  e) Объем данных для сохранения в массив *RcvDat*, указанный в переменной *Size* (объем сохраняемых данных), меньше фактического объема принятых данных.
  f) Модуля, дополнительной платы или порта, указанного с помощью *DevicePort*, не существует.
  g) Недопустимый тип данных переменной *DevicePort*.
  h) Для переменной RcvCfg.EndTrig выбрано значение _SERIAL_END_TERMINATION_CHAR, когда с помощью переменной *DevicePort* выбрана дополнительная плата.
  i) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
  j) Буфер приема заполнен.
  k) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.
  Этой другой командой может быть одна из следующих команд: команда NX_SerialRcv, команда NX_ModbusRtuCmd, команда NX_ModbusRtuRead и команда NX_ModbusRtuWrite.
  l) В принятых данных обнаружена ошибка четности.
  m) В принятых данных обнаружена ошибка кадра.
  n) В принятых данных обнаружена ошибка переполнения.
  o) Истекло время ожидания
  p) Эта команда была выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля.

q) В качестве режима связи по последовательному интерфейсу для указанной дополнительной платы не установлен режим *Без протокола (No-protocol)*.

Пример программы

В данном примере интерфейсный модуль серии NX (NX-CIF210) подключен к интерфейсному модулю EtherCAT (NX-ECC203).
Значение номера модуля, установленное для модуля NX-CIF210: *1*.



Принимаются данные, считанные устройством считывания штрих-кодов, которое подключено к последовательному порту 2 модуля NX-CIF210.
Принятые данные сохраняются во входную-выходную переменную *RecvDat*. Код начала данных не используется. Используется код конца данных: 16#OD (CR).

В следующей таблице приведены значения параметров модуля NX-CIF210.

Параметр	Заданное значение
Порт 2: скорость передачи данных	38 400 бит/с
Порт 2: длина данных	8 бит
Порт 2: проверка четности	Нет
Порт 2: количество стоп-битов	1 бит
Порт 2: управлением потоком данных	Нет

Определения глобальных переменных

● Глобальные переменные

Имя	Тип данных	AT	Комментарий
E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	ECAT://node#1/NX Unit I/O Data Active Status 125	Использование данных ввода-вывода для 63 модулей NX.
N1_Node_location_information	_sNXUNIT_ID	---	Переменная устройства для указания модуля NX-CIF210*1

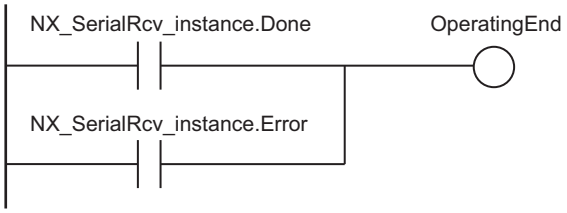
*1. В Sysmac Studio щелкните правой кнопкой мыши ведомый терминал серии NX, выберите **Display Node Location Port (Отобразить порт расположения узла)** и задайте переменную устройства. Дополнительные сведения см. в: *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Программа на языке LD

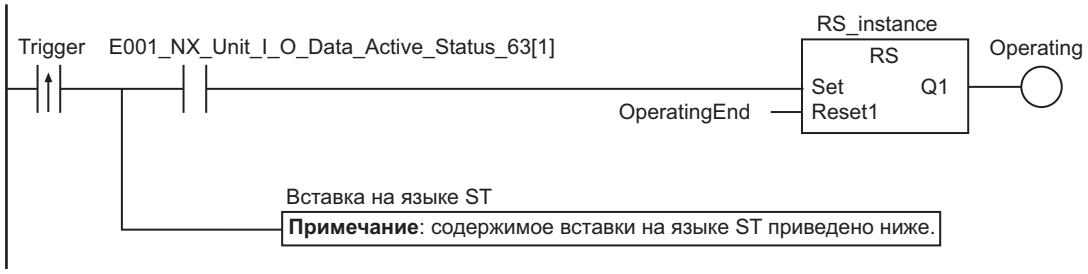
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperationEnd	BOOL	ЛОЖЬ	Обработка завершена
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	DevicePort	_sDEVICE_PORT		Параметры порта
	RecvDat	ARRAY[0..255] OF BYTE	[256(16#0)]	Принимаемые данные
	RecvSize	UINT	0	Объем принятых данных
	RecvStringDat	STRING[257]	''	
	Code	ULINT	0	Штрих-код (целое число)
	RS_instance	RS		
	NX_SerialRcv_instance	NX_SerialRcv		
	RcvCfg	_sSERIAL_CFG		Настройка завершения приема
	StartTrig	_eSERIAL_START	_SERIAL_START_NONE	Не использовать код начала
	StartCode	BYTE[2]	[2(16#0)]	
	EndTrig	_eSERIAL_END	_SERIAL_END_ENDCODE1	Использовать код конца
	EndCode	BYTE[2]	[16#0D, 16#00]	16#0D(CR)
	RcvSizeCfg	UINT	0	
	Option	_sSERIAL_RCV_OPTION		Дополнительные параметры
	TimeOut	TIME	TIME#0 c	
	LastDatRcv	BOOL	ЛОЖЬ	

Внешние переменные	Переменная	Тип данных	Комментарий
	E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	• Использование данных ввода-вывода для 63 модулей NX. • Если номер соответствующего модуля = *1*, используется E001_NX_Unit_I_O_Data_Active_Status_63[1].
	N1_Node_location_information	_sNXUNIT_ID	Переменная устройства для указания модуля NX-CIF210

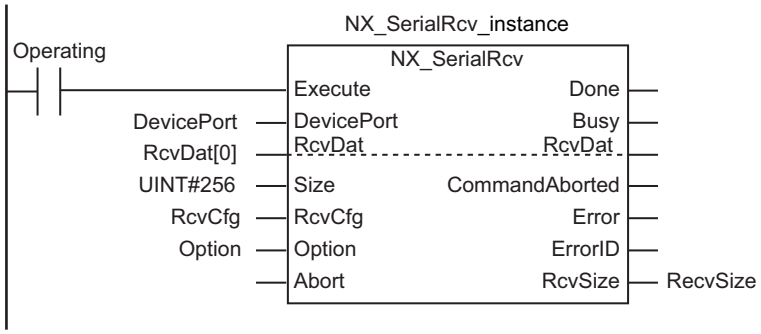
Определение, завершилось ли выполнение команды NX_SerialRcv.



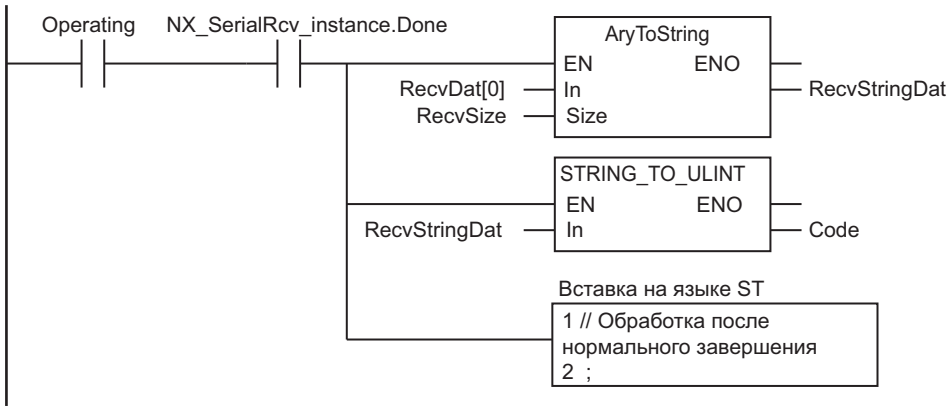
Прием условия выполнения.



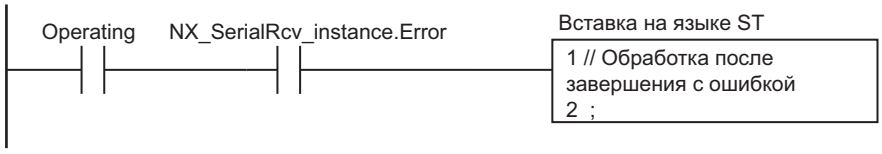
Выполнение команды NX_SerialRcv.



Обработка после нормального завершения



Обработка после завершения с ошибкой



● Содержимое вставки на языке ST

```
DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
DevicePort.NxUnit:=N1_Node_location_information;
DevicePort.PortNo:=2;
```

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась
	Operating	BOOL	ЛОЖЬ	Обработка
	DevicePort	_sDEVICE_PORT		Параметры порта
	RecvDat	ARRAY[0..255] OF BYTE	[256(16#0)]	Принимаемые данные
	RecvSize	UINT	0	Объем принятых данных
	RecvStringDat	STRING[257]	''	
	Code	ULINT	0	Штрих-код (целое число)
	NX_SerialRcv_instance	NX_SerialRcv		
	RcvCfg	_sSERIAL_CFG		Настройка завершения приема
	StartTrig	_eSERIAL_START	_SERIAL_START_NONE	Не использовать код начала
	StartCode	BYTE[2]	[2(16#0)]	
	EndTrig	_eSERIAL_END	_SERIAL_END_ENDCODE1	Использовать код конца
	EndCode	BYTE[2]	[16#0D, 16#00]	16#0D(CR)
	RcvSizeCfg	UINT	0	
	Option	_sSERIAL_RCV_OPTION		Дополнительные параметры
	TimeOut	TIME	TIME#0 c	
	LastDatRcv	BOOL	ЛОЖЬ	

Внешние переменные	Переменная	Тип данных	Комментарий
	E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	• Использование данных ввода-вывода для 63 модулей NX. • Если номер соответствующего модуля = *1*, используется E001_NX_Unit_I_O_Data_Active_Status_63[1].
	N1_Node_location_information	_sNXUNIT_ID	Переменная устройства для указания модуля NX-CIF210

```
// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE)
     AND(E001_NX_Unit_I_O_Data_Active_Status_63[1]) AND (SerialRcv_instance.Busy=FA
LSE) ) THEN
          OperatingStart:=TRUE;
```

```
        Operating:=TRUE;
        DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
        DevicePort.NxUnit:=N1_Node_location_information;
        DevicePort.Port.PortNo:=2;
END_IF;
LastTrigger:=Trigger;

// Настройка параметров связи и инициализация команды SerialRcv.
IF (OperatingStart=TRUE) THEN
        NX_SerialRcv_instance(
                        Execute:=FALSE,         // Инициализация экземпляра.
                        DevicePort:=DevicePort, // Параметры порта
                        Size:=UINT#256,,        // Объем принимаемых данных
                        RcvDat:=RecvDat,        // Принятые данные
                        RcvSize=>RecvSize);     // Объем фактически принятых да
нных
        OperatingStart:=FALSE;
END_IF;
// Выполнение команды NX_SerialRcv.
IF (Operating=TRUE) THEN
        NX_SerialRcv_instance(
                        Execute:=TRUE,
                        DevicePort:=DevicePort,
                        Size:=UINT#256,
                        RcvDat:=RecvDat,
                        RcvSize=>RecvSize);
        IF (NX_SerialRcv_instance.Done=TRUE) THEN
            // Обработка после нормального завершения
            RecvStringDat:=AryToString(In:=RecvDat[0],Size:=RecvSize); // Преобраз
ование кодов символов в текстовую строку.
            Code:=STRING_TO_ULINT(RecvDat); // Преобразование текстовой строки в ц
елое число.

            Operating:=FALSE;
        END_IF;
        IF (NX_SerialRcv_instance.Error=TRUE) THEN
                // Обработка после завершения с ошибкой
                Operating:=FALSE;
        END_IF;
END_IF;
```

NX_ModbusRtuCmd

Команда NX_ModbusRtuCmd передает команды общего назначения через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы ведомым устройствам Modbus-RTU с использованием протокола Modbus-RTU.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_ModbusRtuCmd	Передача команды общего назначения по протоколу Modbus RTU	FB	NX_ModbusRtuCmd_instance NX_ModbusRtuCmd Execute — Done DevicePort — Busy SlaveAdr — CommandAborted CmdDat — Error CmdSize — ErrorID RespDat — RespDat Option — ErrorIDEx Abort — RespSize	NX_ModbusRtuCmd_instance(Execute, DevicePort, SlaveAdr, CmdDat, CmdSize, RespDat, Option, Abort, Done, Busy, CommandAborted, Error, ErrorID, ErrorIDEx, RespSize);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
SlaveAdr	Адрес ведомого устройства		Адрес ведомого устройства Modbus-RTU*1	0...247	---	1
CmdDat[] (массив)	Данные команды		Данные команды	Зависит от типа данных.	---	*2
CmdSize	Объем данных команды		Объем данных команды	1...253	Байты	*2*3
Option	Дополнительные параметры		Дополнительные параметры	---	---	---
Abort	Прерывание		Прерывание выполнения команды	Зависит от типа данных.	---	ЛОЖЬ
RespDat[] (массив)	Прочитанные данные	Вход-выход	Переменная, хранящая прочитанные данные	Зависит от типа данных.	---	---
CommandAborted	Завершение по прерыванию	Выход	Завершение по прерыванию	Зависит от типа данных.	---	---
RespSize	Объем принятых данных		Объем принятых данных	1...253	Байты	*4

*1. Можно задать значение *0* для передачи команд ведомым устройствам Modbus-RTU в режиме широковещания.

*2. Если опустить входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.
*3. Следует задать суммарное количество байтов для кода функции и данных команды. Количество байтов для кода функции: 1.
*4. Сохраняется суммарное количество байтов для кода функции и прочитанных данных. Количество байтов для кода функции: 1.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1461.																			
SlaveAdr							ОК													
CmdDat[] (массив)		ОК																		
CmdSize							ОК													
Option	Подробные сведения о структуре _sSERIAL_MODBUSRTU_OPTION см. в разделе *Функция* на стр. 2-1461.																			
Abort	ОК																			
RespDat[] (массив)		ОК																		
CommandAborted	ОК																			
RespSize							ОК													

Функция

Команда NX_ModbusRtuCmd передает команды общего назначения через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы ведомым устройствам Modbus-RTU с использованием протокола Modbus-RTU.
Эта команда программы завершается нормально, если поступает нормальный ответ на переданную команду связи.
Если команда связи передается в режиме широковещания, эта команда программы завершается нормально, не дожидаясь получения ответов от ведомых устройств.

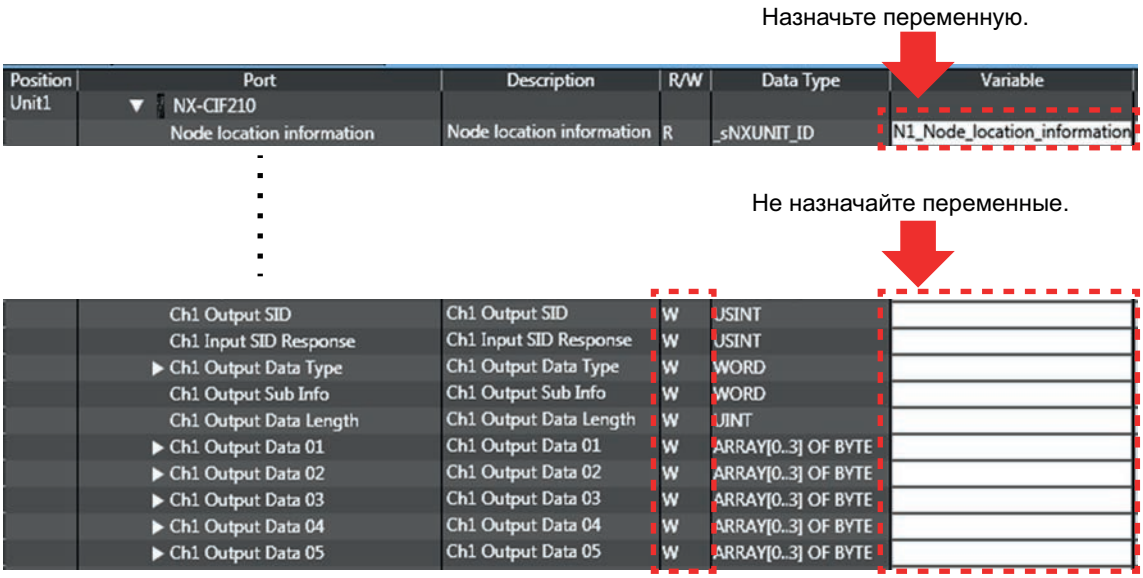
Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.
Задайте значение _DeviceNXUnit для модуля NX или значение _DeviceOptionBoard для дополнительной платы.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

Чтобы указать модуль NX, используйте переменную *NxUnit* для указания устройства.
Переменные *EcatSlave* и *OptBoard* в этом случае не используются.
В переменную *NxUnit* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.
Чтобы указать дополнительную плату, используйте переменную *OptBoard* для указания устройства.
Переменные *NxUnit* и *EcatSlave* в этом случае не используются.
В переменную *OptBoard* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода, следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2
Для модуля NX укажите порт 1 или порт 2.
Для дополнительной платы укажите порт 1.

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceNXUnit или _DeviceOptionBoard.

Используйте входную переменную *SlaveAdr* для указания адреса ведомого устройства Modbus-RTU.
Для передачи команд ведомым устройствам Modbus-RTU в режиме широковещания передайте во входную переменную *SlaveAdr* значение *0*.

Данные команды задаются с помощью входной переменной *CmdDat*, а объем данных команды указывается во входной переменной *CmdSize*.
Код CRC добавляется командой.
Переменная, в которую должны сохраняться прочитанные данные, указывается с помощью входной-выходной переменной *RespDat*.
Фактический объем принятых данных выдается в выходную переменную *RespSize*.

Для настройки дополнительных параметров используется входная переменная *Option*.
Для входной переменной *Option* используется структурный тип данных _sSERIAL_MODBUSRTU_OPTION. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Option	Дополнительные параметры	Дополнительные параметры	_sSERIAL_MODBUSRTU_OPTION	---	---	---
SendDelay	Время задержки передачи	Время задержки передачи в масштабе 1 = 0,01 с.	UINT	Зависит от типа данных.	0,01 с	0
TimeOut	Время ожидания	Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	UINT	Зависит от типа данных.	0,1 с	20
NoResponse	Без ответа	• Задайте значение ИСТИНА, если ожидать ответа на переданную команду не нужно. • Если задано значение ИСТИНА, эта команда программы передает команду связи и завершается нормально, не дожидаясь, пока истечет время ожидания.	BOOL	Зависит от типа данных.	---	ЛОЖЬ
Retry	Количество повторных попыток	Количество повторных попыток	USINT	0...15	---	0

Меры предосторожности для обеспечения надлежащей эксплуатации

Если эта команда будет выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля, произойдет ошибка.

Прерывание выполнения команды

Если вход *Abort* переходит в состояние ИСТИНА во время выполнения команды, выполнение команды прерывается.
Когда выполнение команды прерывается, значение переменной *CommandAborted* меняется на ИСТИНА.
Если выполнение команды уже завершилось к тому моменту, когда делается попытка прервать ее выполнение, значение *Done* меняется на ИСТИНА и команда завершается нормально.
Если входы *Abort* и *Execute* переходят в состояние ИСТИНА одновременно, значение *CommandAborted* меняется на ИСТИНА.

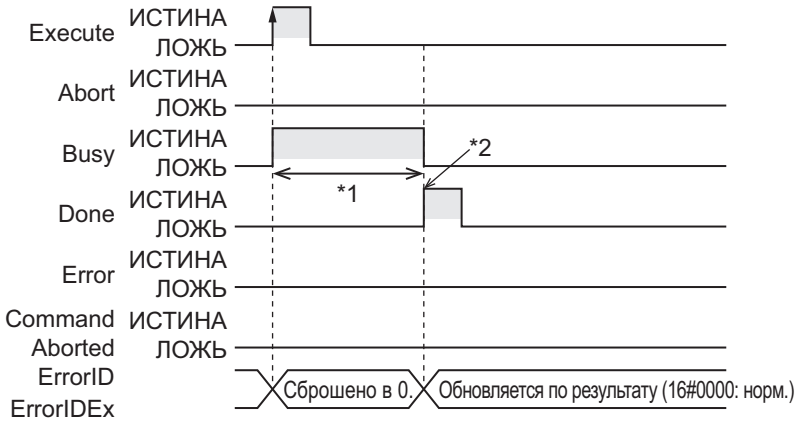
Эта операция прерывания лишь завершает обработку выхода *Busy*, но она не очищает буфер передачи или приема. Для очистки буфера следует использовать команду *NX_SerialBufClear* на стр. 2-1516.

Временные диаграммы

Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение (когда *SendDelay* = *0* (0 с))

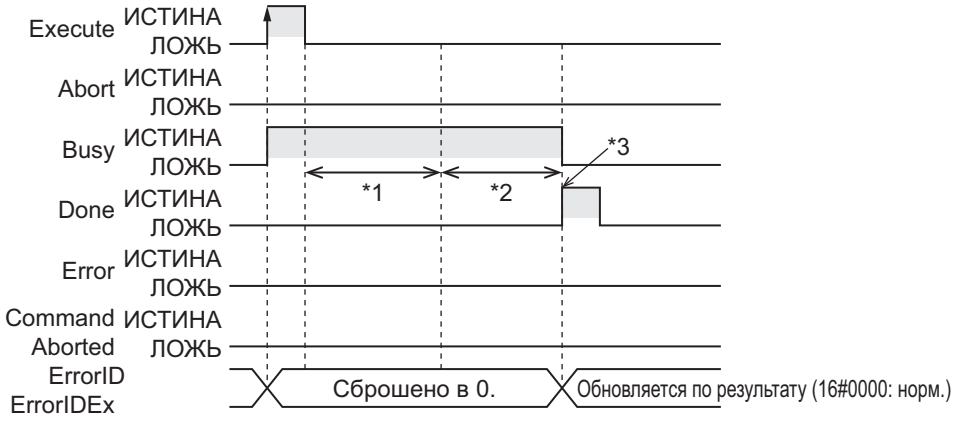
Ниже показана работа команды для случая, когда *SendDelay* = *0* (0 с).



*1. Взаимодействие с ведомым устройством Modbus-RTU
*2. Получен ответ на переданную команду.

● Нормальное завершение (когда *SendDelay* = *100* (1 с))

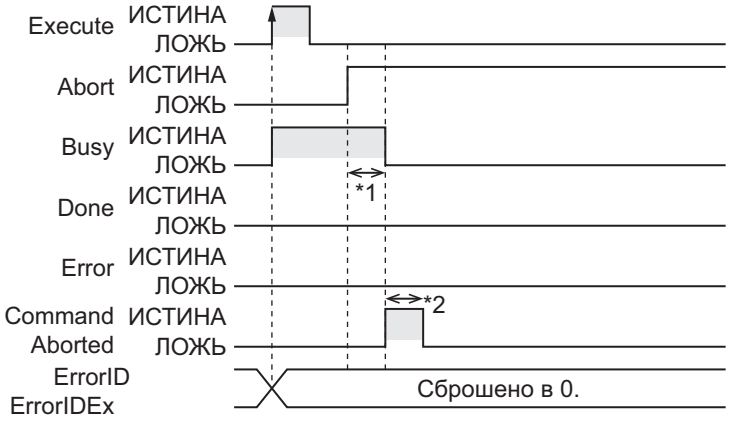
Ниже показана работа команды для случая, когда *SendDelay* = *100* (1 с).



*1. Время задержки передачи = 1 с.
*2. Передается команда ведомому устройству Modbus-RTU, и принимается ответ от ведомого устройства Modbus-RTU.
*3. Получен ответ на переданную команду.

● Выполнено прерывание (когда *Busy* = ИСТИНА)

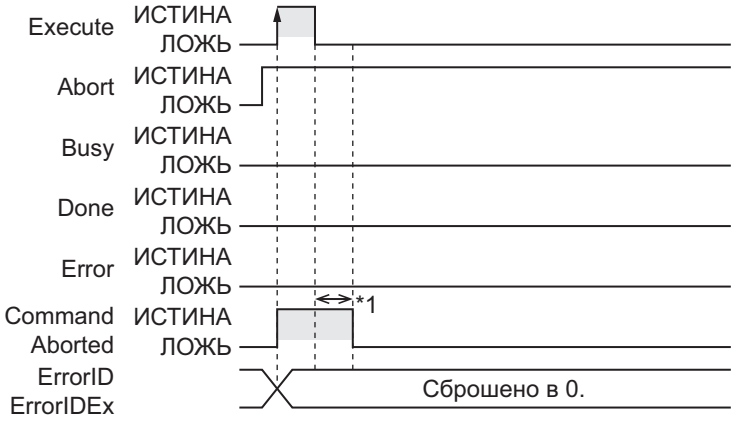
Ниже показана работа команды для случая, когда значение *Abort* меняется на ИСТИНА, когда *Busy* = ИСТИНА.



*1. Обрабатывается прерывание
*2. Меняется на ЛОЖЬ после одного периода выполнения задачи.

● Выполнено прерывание (когда *Execute* = ИСТИНА)

Ниже показана работа команды для случая, когда значения *Abort* и *Execute* меняются на ИСТИНА.



*1. Меняется на ЛОЖЬ после одного периода выполнения задачи.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_OptBoardSta*1	Состояние дополнительной платы	ARRAY[1..2] of _sOPTBOARD_STA	Хранит данные о состоянии дополнительной платы.
_NXB_UnitIOActiveTbl*2	Состояние активности данных ввода-вывода модуля NX	ARRAY[0..32] OF BOOL*3	• Эти данные о состоянии сообщают, может ли тот или иной модуль NX участвовать в обмене данными ввода-вывода. • Индекс элемента массива соответствует номеру модуля NX. Индексу 0 соответствует ведущее устройство шины NX.

*1. Эту переменную можно использовать только с модулями ЦПУ NX1P2.
*2. Эту переменную можно использовать только с модулями ЦПУ NX102 и NX1P2.
*3. Тип данных для модулей ЦПУ NX1P2: *ARRAY [0..8] OF BOOL*.

Дополнительная информация

Ниже показан формат кадра, используемый в режиме Modbus-RTU.

Адрес ведомого устройства	Код функции	Данные	CRC
1 байт	1 байт	0...252 байта	2 байта*

* В коде CRC младший байт идет первым, а старший — вторым.

Описание протокола передачи данных MODBUS см. в документе *MODBUS Application Protocol Specification (Спецификация протокола прикладного уровня MODBUS)*.
Для получения документа *MODBUS Application Protocol Specification* следует обращаться в организацию Modbus Organization, Inc.
(http://www.modbus.org/)

Меры предосторожности для обеспечения надлежащей эксплуатации

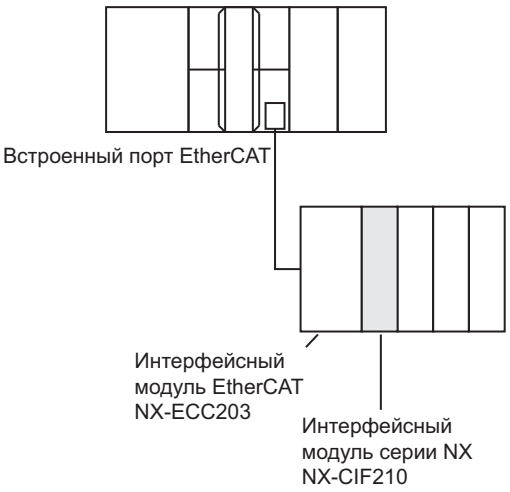
- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи.
  По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки. Если *Abort* перейдет в состояние ИСТИНА во время выполнения команды, значение *CommandAborted* или *Done* поменяется на ИСТИНА.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
- В указанных ниже случаях в буфере адресуемого порта устройства могут оставаться данные. Прежде чем выполнять следующие команды, необходимо выполнить команду NX_SerialBufClear для очистки буфера:
  NX_ModbusRtuCmd instruction, NX_ModbusRtuRead instruction или NX_ModbusRtuWrite instruction.
  a) В начале работы контроллера или после его перехода в режим работы «Выполнение».
  b) При предыдущем выполнении команды было задано выполнение повторных попыток (т. е. значение *Option.Retry* не равно 0).
  c) Предыдущее выполнение команды было прервано (т. е. выходная переменная *CommandAborted* = ИСТИНА).
  d) При предыдущем выполнении команды произошла ошибка (т. е. *Error* = ИСТИНА).
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для переменной *CmdSize*, *Option.Retry*, *DevicePort.DevicePortType*, *DevicePort.PortNo* или *SlaveAdr* задано значение вне допустимого диапазона.
  b) Размер переменной, указанной для *CmdDat*, меньше размера, указанного в *CmdSize*.
  c) Объем принятых данных больше, чем размер переменной, указанной для *RespDat*.
  d) Модуля или порта, указанного с помощью *DevicePort*, не существует.
  e) Недопустимый тип данных переменной *DevicePort*.

f) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
g) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.
Этой другой командой может быть одна из следующих команд: команда NX_SerialSend, команда NX_SerialRcv, команда NX_ModbusRtuCmd, команда NX_ModbusRtuRead и команда NX_ModbusRtuWrite.
h) В принятых данных обнаружена ошибка четности.
i) В принятых данных обнаружена ошибка кадра.
j) В принятых данных обнаружена ошибка переполнения.
k) В принятых данных обнаружено несоответствие контрольной суммы.
l) Истекло время ожидания
m) Эта команда была выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля.
n) От ведомого устройства Modbus-RTU получен ответ с кодом исключения. Значение кода исключения можно проверить в выходной переменной *ErrorIDEx*.
o) В данных ответа от ведомого устройства Modbus-RTU обнаружен недопустимый код функции, неверный объем принятых данных и т. п.
p) В качестве режима связи по последовательному интерфейсу для указанной дополнительной платы не установлен режим *Ведущее устройство Modbus-RTU (Modbus-RTU master)*.

- Данная команда предусматривает отображение дополнительного кода ошибки в переменной *ErrorIDEx* в случае обнаружения ошибки в ведомом устройстве Modbus-RTU. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = *WORD#16#0C10*.
Формат отображения: *ErrorIDEx=000000XX*. Значение *XX* см. в описании кодов исключения протокола передачи данных MODBUS.
Описание кодов исключения протокола передачи данных MODBUS см. в документе *MODBUS Application Protocol Specification (Спецификация протокола прикладного уровня MODBUS)*.
Для получения документа *MODBUS Application Protocol Specification* следует обращаться в организацию Modbus Organization, Inc.
(http://www.modbus.org/)

Пример программы

В данном примере интерфейсный модуль серии NX (NX-CIF210) подключен к интерфейсному модулю EtherCAT (NX-ECC203).
Значение номера модуля, установленное для модуля NX-CIF210: *1*.
В рабочих параметрах модуля для модуля NX-CIF210: задайте параметр **Ch2 Number of Characters to Determine the End (канал 2, число символов для определения конца данных)** равным *35*. Количество символов во время работы принимается за 3,5, поскольку значение параметра Number of Characters to Determine the End (Число символов для определения конца данных) задается в масштабе 1 = 0,1 символа.



Когда переменная *Trigger* принимает значение ИСТИНА, команда программы очищает буфер последовательного порта 2 модуля NX-CIF210, после чего передает команду Modbus-RTU. Считывается содержимое регистра хранения по начальному адресу чтения 32 (BYTE#16#0020) в ведомом устройстве с адресом 1.
Для чтения значения переменной передаются/принимаются команды общего назначения.

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Stage	INT	0	
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	DevicePort	_sDEVICE_PORT		Параметры порта
	NX_SerialBufClear_instance	NX_SerialBufClear		Очистка буфера
	ClearDone	BOOL		
	ClearError	BOOL		
	NX_ModbusRtuCmd_instance	NX_ModbusRtuCmd		
	ModbusSlaveAdr	UINT	UINT#0	Адрес ведомого устройства
	ModbusCmdDat	ARRAY[0..19] OF BYTE		Данные команды Modbus
	ModbusDatSize	UINT	UINT#0	Общий объем данных команды Modbus (байт)
	ModbusRespDat	ARRAY[0..275] OF BYTE		Область хранения принятых данных
	ModbusDone	BOOL		
	ModbusCommandAborted	BOOL		
	ModbusError	BOOL		
	ModbusRspSize	UINT		Фактический объем принятых данных (байт)
	DoModbusTrigger	BOOL		

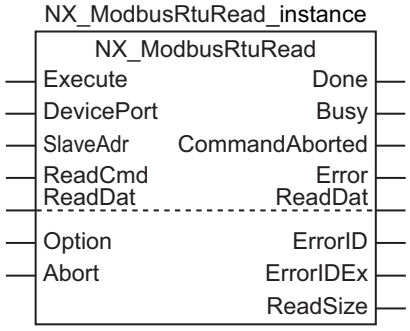
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	N1_Node_location_information	_sNXUNIT_ID	☑	

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF (Trigger=TRUE) AND (DoModbusTrigger=FALSE) THEN
    DoModbusTrigger := TRUE;

    NX_SerialBufClear_instance(Execute := FALSE,
        DevicePort:=DevicePort );
    NX_ModbusRtuCmd_instance(Execute:= FALSE,
        DevicePort:=DevicePort,
        CmdDat:=ModbusCmdDat[1],
        CmdSize:=ModbusDatSize,
        RespDat:=ModbusRespDat[0] );
    Stage := 1; // Инициализация завершена.
END_IF;

IF (DoModbusTrigger=TRUE) THEN
    CASE Stage OF
    1: // Запрос на очистку буфера
        DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
        DevicePort.NxUnit:=N1_Node_location_information;
        DevicePort.PortNo:=2;

        NX_SerialBufClear_instance(Execute := TRUE,
            DevicePort:=DevicePort,
            Done => ClearDone,
            Error => ClearError);

        IF (ClearDone = TRUE) THEN
            Stage := 2; // Очистка буфера нормально завершена.
        ELSIF ( ClearError = TRUE ) THEN
            Stage := 99; // Очистка буфера завершена с ошибкой.
        END_IF;

    2: // Запрос на передачу команды Modbus
        ModbusSlaveAdr := 1; // Адрес ведомого устройства
        ModbusCmdDat[1]:=BYTE#16#03; // Код функции (чтение переменной)
        ModbusCmdDat[2]:=BYTE#16#00; // Начальный адрес чтения (старш.)
        ModbusCmdDat[3]:=BYTE#16#20; // Начальный адрес чтения (младш.)
        ModbusCmdDat[4]:=BYTE#16#00; // Количество данных (старш.)
        ModbusCmdDat[5]:=BYTE#16#01; // Количество данных (младш.)
        ModbusDatSize:=5;

        NX_ModbusRtuCmd_instance(Execute:= TRUE,
            DevicePort:=DevicePort,
            SlaveAdr:=ModbusSlaveAdr,
```

```
            CmdDat:=ModbusCmdDat[1],
            CmdSize:=ModbusDatSize,
            RespDat:=ModbusRespDat[0],
            Done=>ModbusDone,
            CommandAborted=>ModbusCommandAborted,
            Error=>ModbusError,
            RespSize=>ModbusRspSize);

        IF (ModbusDone = TRUE) THEN
            Stage := 3; // Команда NX_ModbusRtuCmd завершена нормально.
        ELSIF (ModbusError=TRUE) OR (ModbusCommandAborted=TRUE) THEN
            Stage :=99; // Команда NX_ModbusRtuCmd завершена с ошибкой или по Abort
.
        END_IF;

    3: // Обработка после нормального завершения команды NX_ModbusRtuCmd.
        Trigger := FALSE;
        DoModbusTrigger := FALSE;

    99: // Обработка ошибки
        Trigger := FALSE;
        DoModbusTrigger := FALSE;
    END_CASE;
END_IF;
```

NX_ModbusRtuRead

Команда NX_ModbusRtuRead передает команды чтения через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы ведомым устройствам Modbus-RTU с использованием протокола Modbus-RTU.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_ModbusRtuRead	Передача команды чтения по протоколу Modbus RTU	FB		NX_ModbusRtuRead_instance(Execute, DevicePort, SlaveAdr, ReadCmd, ReadDat, Option, Abort, Done, Busy, CommandAborted, Error, ErrorID, ErrorIDEx, ReadSize);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
SlaveAdr	Адрес ведомого устройства		Адрес ведомого устройства Modbus-RTU*1	1...247	---	1
ReadCmd	Команда чтения		Команда чтения	---	---	*2
Option	Дополнительные параметры		Дополнительные параметры	---	---	---
Abort	Прерывание		Прерывание выполнения команды	Зависит от типа данных.	---	ЛОЖЬ
ReadDat[] (массив)	Прочитанные данные	Вход-выход	Переменная, хранящая прочитанные данные	Зависит от типа данных.	---	---
CommandAborted	Завершение по прерыванию	Выход	Завершение по прерыванию	Зависит от типа данных.	---	---
ReadSize	Объем принятых данных		Объем принятых данных	1...2000*3	---*4	---

*1. Если будет задано значение *0*, произойдет ошибка.
*2. Если опустить входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.
*3. Верхнее предельное значение в случае приема данных типа WORD: *125*.
*4. Используются те же единицы, что и единицы значения, указанного в *ReadCmd.Fun*.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1473.																			
SlaveAdr							ОК													
ReadCmd	Подробные сведения о структуре _sSERIAL_MODBUSRTU_READ см. в разделе *Функция* на стр. 2-1473.																			
Option	Подробные сведения о структуре _sSERIAL_MODBUSRTU_OPTION см. в разделе *Функция* на стр. 2-1473.																			
Abort	ОК																			
ReadDat[] (массив)	ОК		ОК																	
	Также можно указать массив.																			
CommandAborted	ОК																			
ReadSize							ОК													

Функция

Команда NX_ModbusRtuRead передает команды чтения через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы ведомым устройствам Modbus-RTU с использованием протокола Modbus-RTU. Производится чтение запрошенных данных из ведомых устройств Modbus-RTU.
Эта команда программы завершается нормально, если поступает нормальный ответ на переданную команду связи.

Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

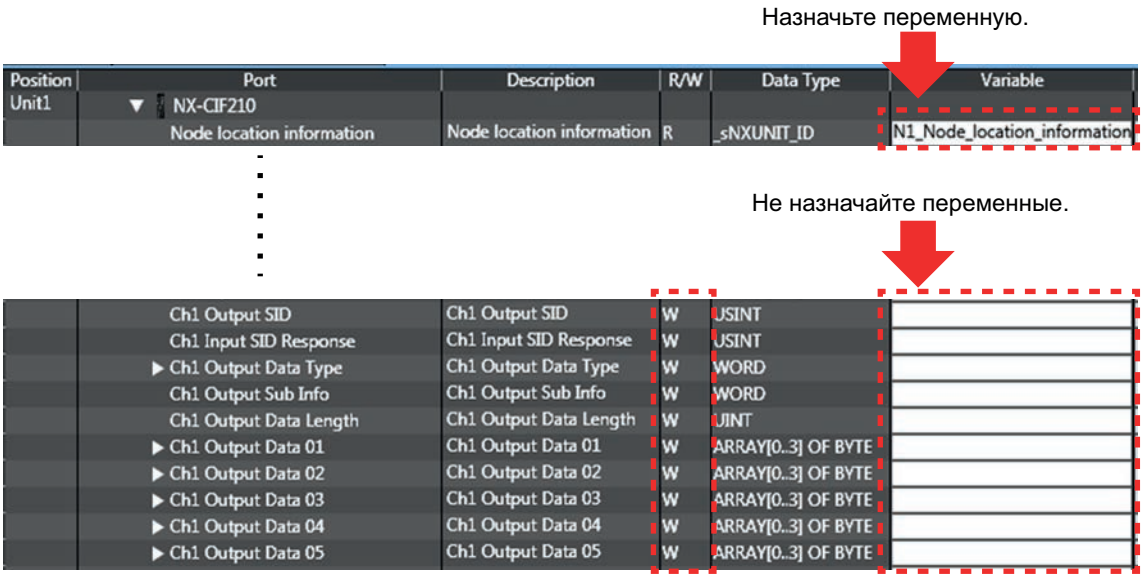
Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.
Задайте значение _DeviceNXUnit для модуля NX или значение _DeviceOptionBoard для дополнительной платы.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

Чтобы указать модуль NX, используйте переменную *NxUnit* для указания устройства.
Переменные *EcatSlave* и *OptBoard* в этом случае не используются.
В переменную *NxUnit* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Чтобы указать дополнительную плату, используйте переменную *OptBoard* для указания устройства.
Переменные *NxUnit* и *EcatSlave* в этом случае не используются.
В переменную *OptBoard* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода, следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2
Для модуля NX укажите порт 1 или порт 2.
Для дополнительной платы укажите порт 1.

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceNXUnit или _DeviceOptionBoard.

Используйте входную переменную *SlaveAdr* для указания адреса ведомого устройства Modbus-RTU.
Если для входной переменной *SlaveAdr* будет задано значение *0*, произойдет ошибка. Передавать команды ведомым устройствам Modbus-RTU в режиме широковещания невозможно.

Используйте входную переменную *ReadCmd* для указания команды чтения.
Код CRC добавляется командой.
Для входной переменной *ReadCmd* используется структурный тип данных _sSERIAL_MODBUSRTU_READ. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
ReadCmd	Команда чтения	Команда чтения	_sSERIAL_MODBUSRTU_READ	---	---	---
Fun	Код функции	Код функции	_eMDB_FUN	_MDB_READ_COILS _MDB_READ_DISCRETE_INPUTS _MDB_READ_HOLDING_REGISTERS _MDB_READ_INPUT_REGISTERS	---	_MDB_READ_COILS
ReadAdr	Адрес для чтения	Начальный адрес чтения	UINT	Зависит от типа данных.	---	0
ReadSize	Размер чтения	Размер чтения	UINT	Зависит от кода функции.	---*1	1

*1. Используются те же единицы, что и единицы значения, указанного в *ReadCmd.Fun*.

Для переменной *Fun* используется перечислимый тип данных _eMDB_FUN.
Значения перечислителей перечислимого типа _eMDB_FUN приведены в таблице ниже:

Перечислитель	Значение
_MDB_READ_COILS	Чтение выходов (бит)
_MDB_READ_DISCRETE_INPUTS	Чтение входов (бит)
_MDB_READ_HOLDING_REGISTERS	Чтение регистров хранения (слово)
_MDB_READ_INPUT_REGISTERS	Чтение входных регистров (слово)

Диапазон допустимых значений переменной *ReadSize* (т. е. объем прочитанных данных) варьируется в зависимости от кода функции.
Каждое значение определяется объемом прочитанных данных и максимальной длиной команды.
Описание приведено в таблице ниже.

Код функции	ReadSize
_MDB_READ_COILS	1...2000 (бит)
_MDB_READ_DISCRETE_INPUTS	1...2000 (бит)
_MDB_READ_HOLDING_REGISTERS	1...125 (слов)
_MDB_READ_INPUT_REGISTERS	1...125 (слов)

Переменная, в которую должны сохраняться прочитанные данные, указывается с помощью входной-выходной переменной *ReadDat*.
Тип данных, который можно использовать для переменной *ReadDat*, зависит от кода функции.
Описание приведено в таблице ниже.

Код функции	Тип данных
_MDB_READ_COILS	BOOL BOOL[]
_MDB_READ_DISCRETE_INPUTS	BOOL BOOL[]
_MDB_READ_HOLDING_REGISTERS	WORD WORD[]
_MDB_READ_INPUT_REGISTERS	WORD WORD[]

Фактический объем прочитанных данных выдается в выходную переменную *ReadSize*.

Для настройки дополнительных параметров используется входная переменная *Option*.
Для входной переменной *Option* используется структурный тип данных _sSERIAL_MODBUSRTU_OPTION. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Option	Дополнительные параметры	Дополнительные параметры	_sSERIAL_MODBUSRTU_OPTION	---	---	---
SendDelay	Время задержки передачи	Время задержки передачи	UINT	Зависит от типа данных.	0,01 с	0
TimeOut	Время ожидания	Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	UINT	Зависит от типа данных.	0,1 с	20
NoResponse	Без ответа	В этой команде не используется.	BOOL	Зависит от типа данных.	---	ЛОЖЬ
Retry	Количество повторных попыток	Количество повторных попыток	USINT	0...15	---	0

Меры предосторожности для обеспечения надлежащей эксплуатации

Если эта команда будет выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля, произойдет ошибка.

Прерывание выполнения команды

Если вход *Abort* переходит в состояние ИСТИНА во время выполнения команды, выполнение команды прерывается.
Когда выполнение команды прерывается, значение переменной *CommandAborted* меняется на ИСТИНА.
Если выполнение команды уже завершилось к тому моменту, когда делается попытка прервать ее выполнение, значение *Done* меняется на ИСТИНА и команда завершается нормально.
Если входы *Abort* и *Execute* переходят в состояние ИСТИНА одновременно, значение *CommandAborted* меняется на ИСТИНА.

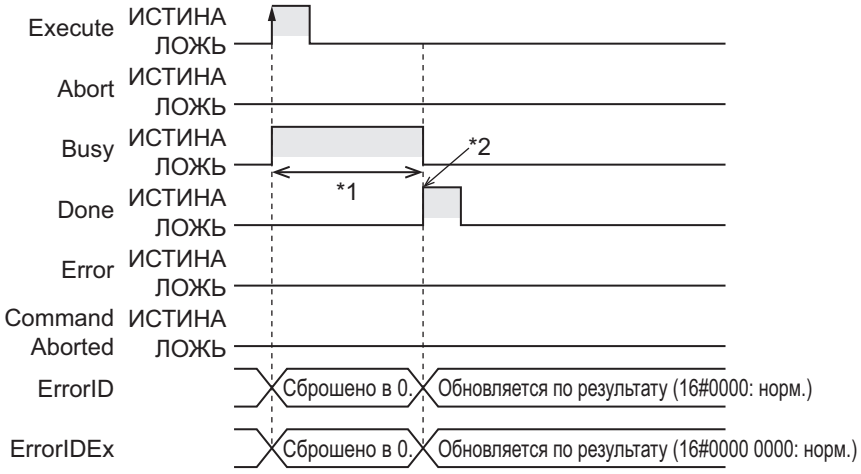
Эта операция прерывания лишь завершает обработку выхода *Busy*, но она не очищает буфер передачи или приема. Для очистки буфера следует использовать команду *NX_SerialBufClear* на стр. 2-1516.

Временные диаграммы

Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение (когда *SendDelay* = *0* (0 с))

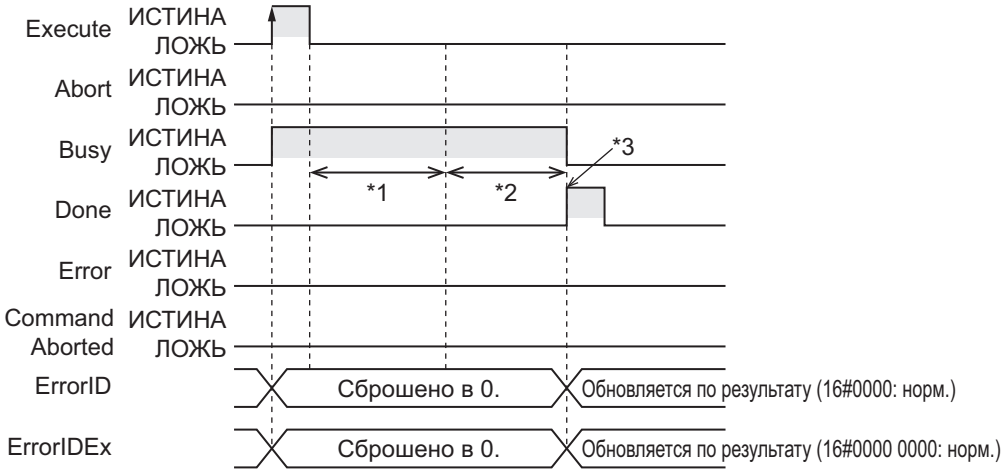
Ниже показана работа команды для случая, когда *SendDelay* = *0* (0 с).



*1. Взаимодействие с ведомым устройством Modbus-RTU
*2. Получен ответ на переданную команду.

● Нормальное завершение (когда *SendDelay* = *100* (1 с))

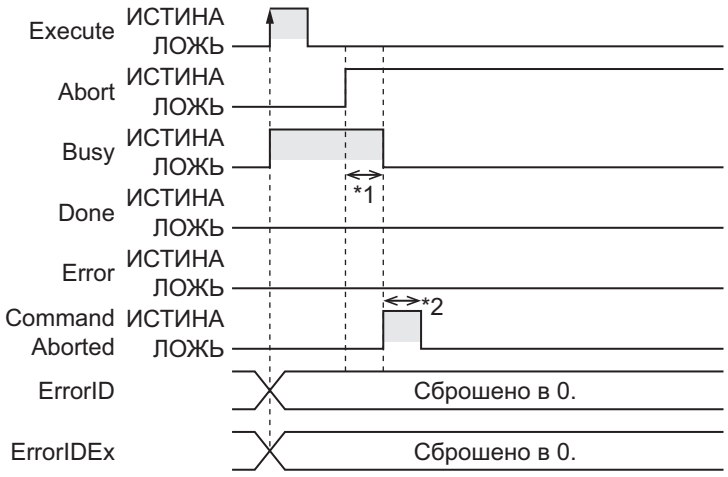
Ниже показана работа команды для случая, когда *SendDelay* = *100* (1 с).



*1. Время задержки передачи = 1 с.
*2. Передается команда чтения ведомому устройству Modbus-RTU, и принимается ответ от ведомого устройства Modbus-RTU.
*3. Получен ответ на переданную команду.

● Выполнено прерывание (когда *Busy* = ИСТИНА)

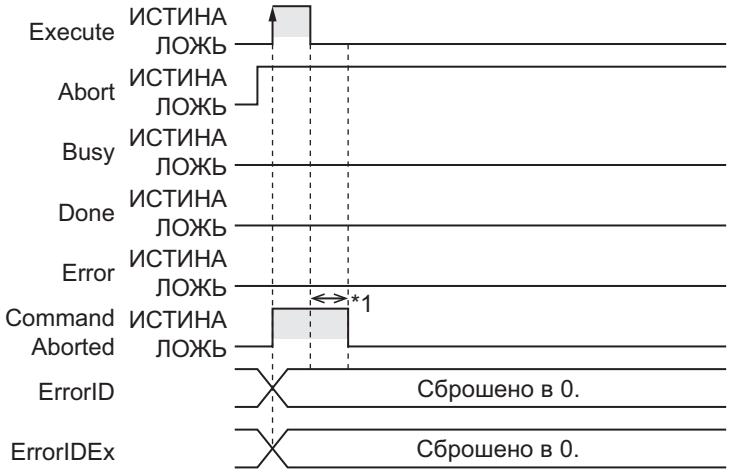
Ниже показана работа команды для случая, когда значение *Abort* меняется на ИСТИНА, когда *Busy* = ИСТИНА.



*1. Обрабатывается прерывание
*2. Меняется на ЛОЖЬ после одного периода выполнения задачи.

● Выполнено прерывание (когда *Execute* = ИСТИНА)

Ниже показана работа команды для случая, когда значения *Abort* и *Execute* меняются на ИСТИНА.



*1. Меняется на ЛОЖЬ после одного периода выполнения задачи.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_OptBoardSta*1	Состояние дополнительной платы	ARRAY[1..2] of _sOPTBOARD_STA	Хранит данные о состоянии дополнительной платы.
_NXB_UnitIOActiveTbl*2	Состояние активности данных ввода-вывода модуля NX	ARRAY[0..32] OF BOOL*3	• Эти данные о состоянии сообщают, может ли тот или иной модуль NX участвовать в обмене данными ввода-вывода. • Индекс элемента массива соответствует номеру модуля NX. Индексу 0 соответствует ведущее устройство шины NX.

*1. Эту переменную можно использовать только с модулями ЦПУ NX1P2.
*2. Эту переменную можно использовать только с модулями ЦПУ NX102 и NX1P2.
*3. Тип данных для модулей ЦПУ NX1P2: *ARRAY [0..8] OF BOOL*.

Дополнительная информация

Описание протокола передачи данных MODBUS см. в документе *MODBUS Application Protocol Specification (Спецификация протокола прикладного уровня MODBUS)*.
Для получения документа *MODBUS Application Protocol Specification* следует обращаться в организацию Modbus Organization, Inc.
(http://www.modbus.org/)

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи.
  По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки. Если *Abort* перейдет в состояние ИСТИНА во время выполнения команды, значение *CommandAborted* или *Done* поменяется на ИСТИНА.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
- В указанных ниже случаях в буфере адресуемого порта устройства могут оставаться данные. Прежде чем выполнять следующие команды, необходимо выполнить команду NX_SerialBufClear для очистки буфера:
  NX_ModbusRtuCmd instruction, NX_ModbusRtuRead instruction или NX_ModbusRtuWrite instruction.
  a) В начале работы контроллера или после его перехода в режим работы «Выполнение».
  b) При предыдущем выполнении команды было задано выполнение повторных попыток (т. е. значение *Option.Retry* не равно 0).
  c) Предыдущее выполнение команды было прервано (т. е. выходная переменная *CommandAborted* = ИСТИНА).
  d) При предыдущем выполнении команды произошла ошибка (т. е. *Error* = ИСТИНА).
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для переменной *SlaveAdr*, *ReadCmd.ReadSize*, *ReadCmd.Fun*, *Option.Retry*, *DevicePort.DevicePortType* или *DevicePort.PortNo* задано значение вне допустимого диапазона.
  b) Размер переменной, указанной для *ReadDat*, меньше размера, указанного в *ReadCmd.ReadSize*.
  c) Модуля или порта, указанного с помощью *DevicePort*, не существует.
  d) Недопустимый тип данных переменной *DevicePort* или *RespDat*.
  e) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
  f) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.

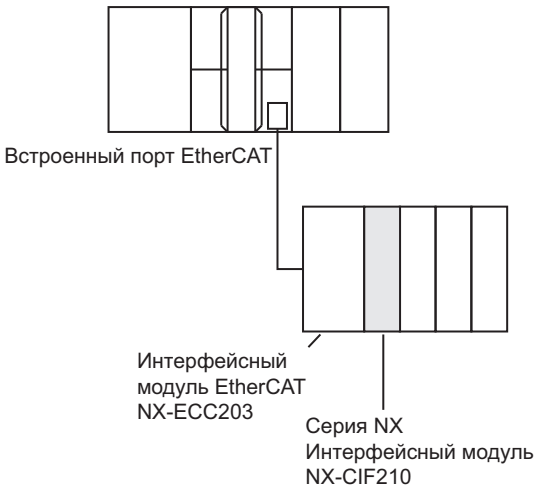
Этой другой командой может быть одна из следующих команд: команда NX_SerialSend, команда NX_SerialRcv, команда NX_ModbusRtuCmd, команда NX_ModbusRtuRead и команда NX_ModbusRtuWrite.

g) В принятых данных обнаружена ошибка четности.
h) В принятых данных обнаружена ошибка кадра.
i) В принятых данных обнаружена ошибка переполнения.
j) В принятых данных обнаружено несоответствие контрольной суммы.
k) Истекло время ожидания (если заданы повторные попытки, время ожидания умножается на количество повторных попыток).
l) Эта команда была выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля.
m) От ведомого устройства Modbus-RTU получен ответ с кодом исключения. Значение кода исключения можно проверить в выходной переменной *ErrorIDEx*.
n) В данных ответа от ведомого устройства Modbus-RTU обнаружен недопустимый код функции, неверный объем принятых данных и т. п.
o) В качестве режима связи по последовательному интерфейсу для указанной дополнительной платы не установлен режим *Ведущее устройство Modbus-RTU (Modbus-RTU master)*.

- Данная команда предусматривает отображение дополнительного кода ошибки в переменной *ErrorIDEx* в случае обнаружения ошибки в ведомом устройстве Modbus-RTU. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID = WORD#16#0C10*. Формат отображения: *ErrorIDEx=000000XX*. Значение *XX* см. в описании кодов исключения протокола передачи данных MODBUS.
  Описание кодов исключения протокола передачи данных MODBUS см. в документе *MODBUS Application Protocol Specification (Спецификация протокола прикладного уровня MODBUS)*. Для получения документа *MODBUS Application Protocol Specification* следует обращаться в организацию Modbus Organization, Inc.
  (http://www.modbus.org/)

Пример программы

В данном примере интерфейсный модуль серии NX (NX-CIF210) подключен к интерфейсному модулю EtherCAT (NX-ECC203).
Значение номера модуля, установленное для модуля NX-CIF210: *1*.
В рабочих параметрах модуля для модуля NX-CIF210: задайте параметр **Ch2 Number of Characters to Determine the End (канал 2, число символов для определения конца данных)** равным *35*. Количество символов во время работы принимается за 3,5, поскольку значение параметра Number of Characters to Determine the End (Число символов для определения конца данных) задается в масштабе 1 = 0,1 символа.



Когда переменная *Trigger* принимает значение ИСТИНА, команда программы очищает буфер последовательного порта 2 модуля NX-CIF210, после чего передает команду Modbus-RTU.
Считывается состояние выхода по начальному адресу чтения 19 в ведомом устройстве с адресом 1.
Для чтения значения переменной передается команда чтения.

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Stage	INT	0	
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	DevicePort	_sDEVICE_PORT		Параметры порта
	NX_SerialBufClear_instance	NX_SerialBufClear		Очистка буфера
	ClearDone	BOOL		
	ClearError	BOOL		
	NX_ModbusRtuRead_instance	NX_ModbusRtuRead		
	ModbusSlaveAdr	UINT	UINT#0	Адрес ведомого устройства
	ModbusDone	BOOL		
	ModbusCommandAborted	BOOL		
	ModbusError	BOOL		
	ModbusReadSize	UINT		Фактический объем принятых данных (байт)
	DoModbusTrigger	BOOL		
	ModbusReadDat	BOOL		
	ModbusReadCmd	_sSERIAL_MODBUSRTU_READ		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	N1_Node_location_information	_sNXUNIT_ID	☑	

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF (Trigger=TRUE) AND (DoModbusTrigger=FALSE) THEN
    DoModbusTrigger := TRUE;

    NX_SerialBufClear_instance(Execute := FALSE,
        DevicePort:=DevicePort);
    NX_ModbusRtuRead_instance(Execute:= FALSE,
        DevicePort:=DevicePort,
        ReadDat:=ModbusReadDat);
    Stage := 1; // Инициализация завершена.
END_IF;

IF (DoModbusTrigger=TRUE) THEN
    CASE Stage OF
    1: // Запрос на очистку буфера
        DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
        DevicePort.NxUnit:=N1_Node_location_information;
        DevicePort.PortNo:=2;

        NX_SerialBufClear_instance(Execute := TRUE,
            DevicePort:=DevicePort,
            Done => ClearDone,
            Error => ClearError);

        IF (ClearDone = TRUE) THEN
            Stage := 2; // Очистка буфера нормально завершена.
        ELSIF (ClearError = TRUE) THEN
            Stage := 99; // Очистка буфера завершена с ошибкой.
        END_IF;

    2: // Запрос чтения по протоколу Modbus
        ModbusSlaveAdr := 1; // Адрес ведомого устройства
        ModbusReadCmd.Fun:=_MDB_READ_COILS; // Код функции
        ModbusReadCmd.ReadAdr:=19; // Адрес чтения
        ModbusReadCmd.ReadSize:=1; // Размер чтения

        NX_ModbusRtuRead_instance(Execute:= TRUE,
            DevicePort:=DevicePort,
            SlaveAdr:=ModbusSlaveAdr,
            ReadCmd:=ModbusReadCmd,
            ReadDat:=ModbusReadDat,
            Done=>ModbusDone,
            CommandAborted=>ModbusCommandAborted,
            Error=>ModbusError,
```

```
            ReadSize=>ModbusReadSize);

        IF (ModbusDone = TRUE) THEN
            Stage := 3; // Команда NX_ModbusRead завершена нормально.
        ELSIF (ModbusError=TRUE) OR (ModbusCommandAborted=TRUE) THEN
            Stage :=99; // Команда NX_ModbusRead завершена с ошибкой или по Abort.
        END_IF;

    3: // Обработка после нормального завершения команды NX_ModbusRead.
        Trigger := FALSE;
        DoModbusTrigger := FALSE;

    99: // Обработка ошибки
        Trigger := FALSE;
        DoModbusTrigger := FALSE;
    END_CASE;
END_IF;
```

NX_ModbusRtuWrite

Команда NX_ModbusRtuWrite передает команды записи через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы ведомым устройствам Modbus-RTU с использованием протокола Modbus-RTU.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_ModbusRtuWrite	Передача команды записи по протоколу Modbus RTU	FB	NX_ModbusRtuWrite_instance NX_ModbusRtuWrite Execute — Done DevicePort — Busy SlaveAdr — CommandAborted WriteCmd — Error WriteDat — ErrorID Option — ErrorIDEx Abort	NX_ModbusRtuWrite_instance(Execute, DevicePort, SlaveAdr, WriteCmd, WriteDat, Option, Abort, Done, Busy, CommandAborted, Error, ErrorID, ErrorIDEx);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
SlaveAdr	Адрес ведомого устройства		Адрес ведомого устройства Modbus-RTU*1	0...247	---	1
WriteCmd	Команда записи		Команда записи	---	---	*2
WriteDat[] (массив)	Записываемые данные		Записываемые данные	Зависит от типа данных.	---	*2
Option	Дополнительные параметры		Дополнительные параметры	---	---	---
Abort	Прерывание		Прерывание выполнения команды	Зависит от типа данных.	---	ЛОЖЬ
CommandAborted	Завершение по прерыванию	Выход	Завершение по прерыванию	Зависит от типа данных.	---	---

*1. Можно задать значение *0* для передачи команд ведомым устройствам Modbus-RTU в режиме широковещания.
*2. Если опустить входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1486.																			
SlaveAdr							ОК													
WriteCmd	Подробные сведения о структуре _sSERIAL_MODBUSRTU_WRITE см. в разделе *Функция* на стр. 2-1486.																			
WriteDat[] (массив)	ОК		ОК																	
	Также можно указать массив.																			
Option	Подробные сведения о структуре _sSERIAL_MODBUSRTU_OPTION см. в разделе *Функция* на стр. 2-1486.																			
Abort	ОК																			
CommandAborted	ОК																			

Функция

Команда NX_ModbusRtuWrite передает команды записи через последовательный порт интерфейсного модуля серии NX или последовательный порт дополнительной платы ведомым устройствам Modbus-RTU с использованием протокола Modbus-RTU.
Эта команда программы завершается нормально, если поступает нормальный ответ на переданную команду связи.
Если команда связи передается в режиме широковещания, эта команда программы завершается нормально, не дожидаясь получения ответов от ведомых устройств.

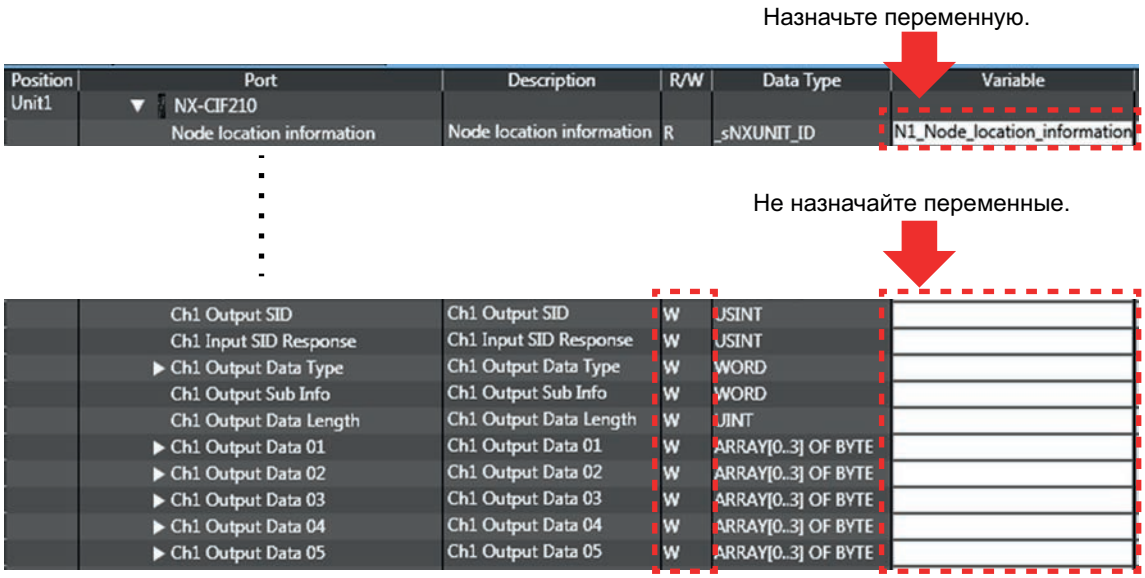
Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.
Задайте значение _DeviceNXUnit для модуля NX или значение _DeviceOptionBoard для дополнительной платы.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

Чтобы указать модуль NX, используйте переменную *NxUnit* для указания устройства.
Переменные *EcatSlave* и *OptBoard* в этом случае не используются.
В переменную *NxUnit* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.
Чтобы указать дополнительную плату, используйте переменную *OptBoard* для указания устройства.
Переменные *NxUnit* и *EcatSlave* в этом случае не используются.
В переменную *OptBoard* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода, следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2
Для модуля NX укажите порт 1 или порт 2.
Для дополнительной платы укажите порт 1.

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceNXUnit или _DeviceOptionBoard.

Используйте входную переменную *SlaveAdr* для указания адреса ведомого устройства Modbus-RTU.
Для передачи команд ведомым устройствам Modbus-RTU в режиме широковещания передайте во входную переменную *SlaveAdr* значение *0*.

Используйте входную переменную *WriteCmd* для указания команды записи.
Код CRC добавляется командой.
Для входной переменной *WriteCmd* используется структурный тип данных _sSERIAL_MODBUSRTU_WRITE. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
WriteCmd	Команда записи	Команда записи	_sSERIAL_MODBUSRTU_WRITE	---	---	---
Fun	Код функции	Код функции	_eMDB_FUN	_MDB_WRITE_SINGLE_COIL _MDB_WRITE_SINGLE_REGISTER _MDB_WRITE_MULTIPLE_COILS _MDB_WRITE_MULTIPLE_REGISTERS	---	_eMDB_WRITE_SINGLE_COIL
WriteAdr	Адрес для записи	Начальный адрес записи	UINT	Зависит от типа данных.	---	0
WriteSize	Размер записи	Размер записи	UINT	Зависит от кода функции.	---	_MDB_WRITE_SINGLE_COIL

Для переменной *Fun* используется перечислимый тип данных _eMDB_FUN.
Значения перечислителей перечислимого типа _eMDB_FUN приведены в таблице ниже:

Перечислитель	Значение
_MDB_WRITE_SINGLE_COIL	Запись в выход (бит)
_MDB_WRITE_SINGLE_REGISTER	Запись в регистр хранения (слово)
_MDB_WRITE_MULTIPLE_COILS	Запись в несколько выходов (бит)
_MDB_WRITE_MULTIPLE_REGISTERS	Запись в несколько регистров хранения (слово)

Диапазон допустимых значений переменной *WriteSize* (т. е. объем записываемых данных) варьируется в зависимости от кода функции.
Каждое значение определяется объемом записываемых данных и максимальной длиной команды.
Описание приведено в таблице ниже.

Код функции	WriteSize
_MDB_WRITE_SINGLE_COIL	1 (бит)
_MDB_WRITE_SINGLE_REGISTER	1 (слово)
_MDB_WRITE_MULTIPLE_COILS	1...1 968 (бит)
_MDB_WRITE_MULTIPLE_REGISTERS	1...123 (слов)

Используйте входную переменную *WriteDat* для указания данных, которые должны быть записаны.
Тип данных, который можно использовать для переменной *WriteDat*, зависит от кода функции.
Описание приведено в таблице ниже.

Код функции	Тип данных
_MDB_WRITE_SINGLE_COIL	BOOL BOOL[]
_MDB_WRITE_SINGLE_REGISTER	WORD WORD[]
_MDB_WRITE_MULTIPLE_COILS	BOOL BOOL[]
_MDB_WRITE_MULTIPLE_REGISTERS	WORD WORD[]

Для настройки дополнительных параметров используется входная переменная *Option*.

Для входной переменной *Option* используется структурный тип данных _sSERIAL_MODBUSRTU_OPTION. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
Option	Дополнительные параметры	Дополнительные параметры	_sSERIAL_MODBUSRTU_OPTION	---	---	---
SendDelay	Время задержки передачи	Время задержки передачи	UINT	Зависит от типа данных.	0,01 с	0
TimeOut	Время ожидания	Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	UINT	Зависит от типа данных.	0,1 с	20
NoResponse	Без ответа	В этой команде не используется.	BOOL	Зависит от типа данных.	---	ЛОЖЬ
Retry	Количество повторных попыток	Количество повторных попыток	USINT	0...15	---	0

Меры предосторожности для обеспечения надлежащей эксплуатации

Если эта команда будет выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля, произойдет ошибка.

Прерывание выполнения команды

Если вход *Abort* переходит в состояние ИСТИНА во время выполнения команды, выполнение команды прерывается.
Когда выполнение команды прерывается, значение переменной *CommandAborted* меняется на ИСТИНА.
Если выполнение команды уже завершилось к тому моменту, когда делается попытка прервать ее выполнение, значение *Done* меняется на ИСТИНА и команда завершается нормально.
Если входы *Abort* и *Execute* переходят в состояние ИСТИНА одновременно, значение *CommandAborted* меняется на ИСТИНА.

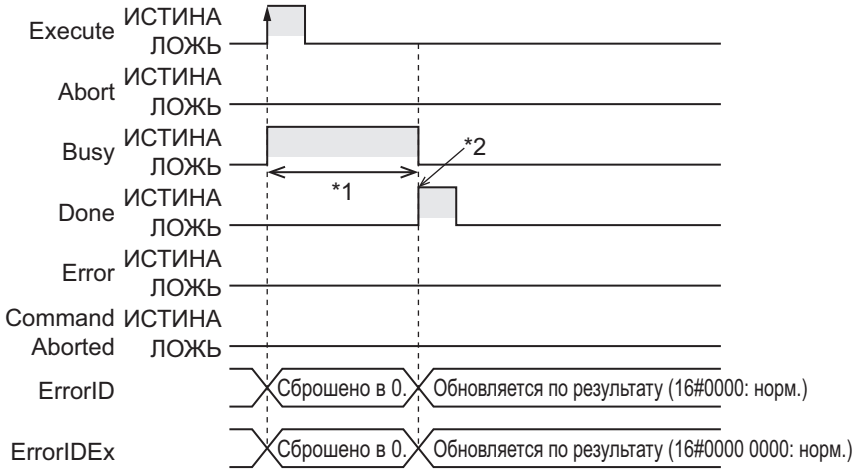
Эта операция прерывания лишь завершает обработку выхода *Busy*, но она не очищает буфер передачи или приема. Для очистки буфера следует использовать команду *NX_SerialBufClear* на стр. 2-1516.

Временные диаграммы

Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение (когда *SendDelay* = *0* (0 с))

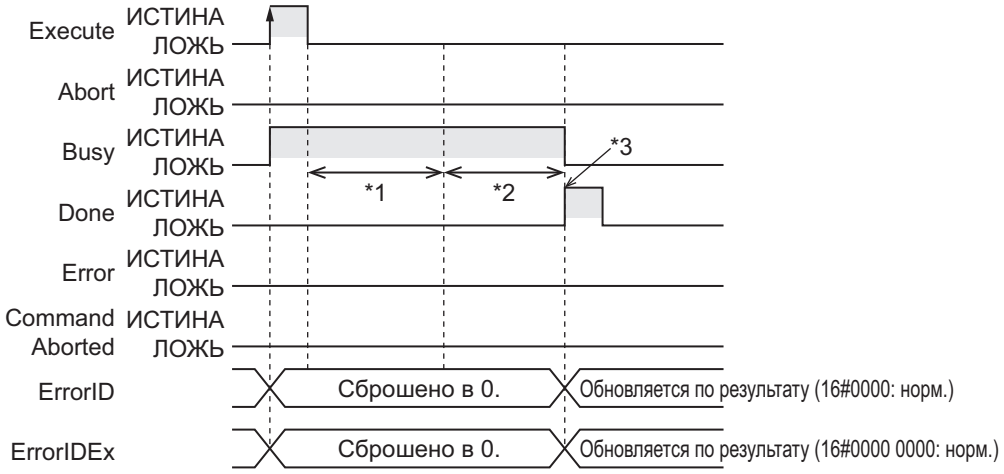
Ниже показана работа команды для случая, когда *SendDelay* = *0* (0 с).



*1. Взаимодействие с ведомым устройством Modbus-RTU
*2. Получен ответ на переданную команду.

● Нормальное завершение (когда *SendDelay* = *100* (1 с))

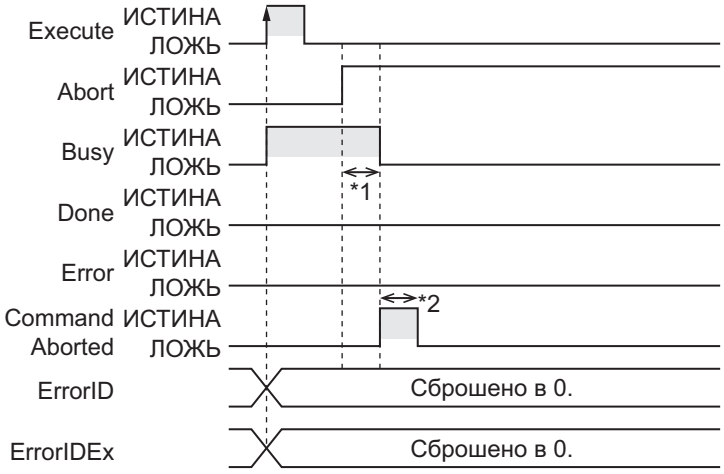
Ниже показана работа команды для случая, когда *SendDelay* = *100* (1 с).



*1. Время задержки передачи = 1 с.
*2. Передается команда записи ведомому устройству Modbus-RTU, и принимается ответ от ведомого устройства Modbus-RTU.
*3. Получен ответ на переданную команду.

● Выполнено прерывание (когда *Busy* = ИСТИНА)

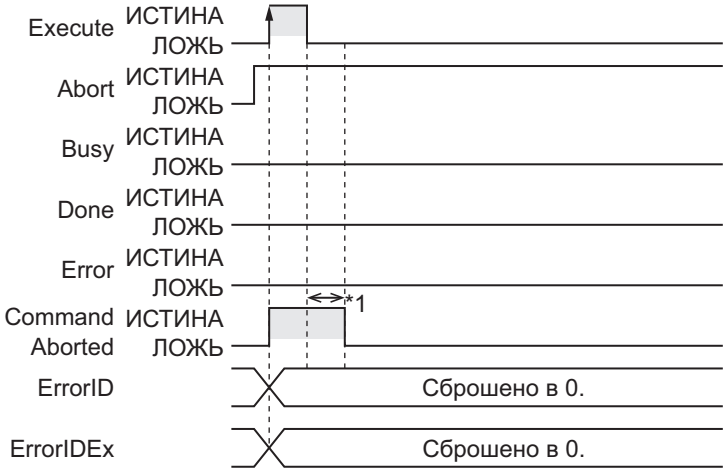
Ниже показана работа команды для случая, когда значение *Abort* меняется на ИСТИНА, когда *Busy* = ИСТИНА.



*1. Обрабатывается прерывание
*2. Меняется на ЛОЖЬ после одного периода выполнения задачи.

● Выполнено прерывание (когда *Execute* = ИСТИНА)

Ниже показана работа команды для случая, когда значения *Abort* и *Execute* меняются на ИСТИНА.



*1. Меняется на ЛОЖЬ после одного периода выполнения задачи.

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_OptBoardSta*1	Состояние дополнительной платы	ARRAY[1..2] of _sOPTBOARD_STA	Хранит данные о состоянии дополнительной платы.
_NXB_UnitIOActiveTbl*2	Состояние активности данных ввода-вывода модуля NX	ARRAY[0..32] OF BOOL*3	• Эти данные о состоянии сообщают, может ли тот или иной модуль NX участвовать в обмене данными ввода-вывода. • Индекс элемента массива соответствует номеру модуля NX. Индексу 0 соответствует ведущее устройство шины NX.

*1. Эту переменную можно использовать только с модулями ЦПУ NX1P2.
*2. Эту переменную можно использовать только с модулями ЦПУ NX102 и NX1P2.
*3. Тип данных для модулей ЦПУ NX1P2: *ARRAY [0..8] OF BOOL*.

Дополнительная информация

Описание протокола передачи данных MODBUS см. в документе *MODBUS Application Protocol Specification (Спецификация протокола прикладного уровня MODBUS)*.
Для получения документа *MODBUS Application Protocol Specification* следует обращаться в организацию Modbus Organization, Inc.
(http://www.modbus.org/)

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи.
  По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки. Если *Abort* перейдет в состояние ИСТИНА во время выполнения команды, значение *CommandAborted* или *Done* поменяется на ИСТИНА.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
- В указанных ниже случаях в буфере адресуемого порта устройства могут оставаться данные. Прежде чем выполнять следующие команды, необходимо выполнить команду NX_SerialBufClear для очистки буфера:
  NX_ModbusRtuCmd instruction, NX_ModbusRtuRead instruction или NX_ModbusRtuWrite instruction.
  a) В начале работы контроллера или после его перехода в режим работы «Выполнение».
  b) При предыдущем выполнении команды было задано выполнение повторных попыток (т. е. значение *Option.Retry* не равно 0).
  c) Предыдущее выполнение команды было прервано (т. е. выходная переменная *CommandAborted* = ИСТИНА).
  d) При предыдущем выполнении команды произошла ошибка (т. е. *Error* = ИСТИНА).
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для переменной *SlaveAdr*, *WriteCmd.Fun*, *WriteCmd.WriteSize*, *Option.Retry*, *DevicePort.DevicePortType* или *DevicePort.PortNo* задано значение вне допустимого диапазона.
  b) Размер переменной, указанной для *WriteDat*, меньше размера, указанного в *WriteCmd.WriteSize*.
  c) Модуля или порта, указанного с помощью *DevicePort*, не существует.
  d) Недопустимый тип данных переменной *DevicePort* или *WriteDat*.
  e) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
  f) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.

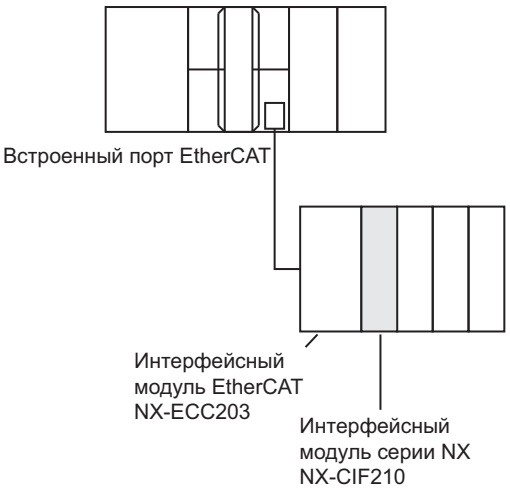
Этой другой командой может быть одна из следующих команд: команда NX_SerialSend, команда NX_SerialRcv, команда NX_ModbusRtuCmd, команда NX_ModbusRtuRead и команда NX_ModbusRtuWrite.

g) В принятых данных обнаружена ошибка четности.
h) В принятых данных обнаружена ошибка кадра.
i) В принятых данных обнаружена ошибка переполнения.
j) В принятых данных обнаружено несоответствие контрольной суммы.
k) Истекло время ожидания
l) Эта команда была выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля.
m) От ведомого устройства Modbus-RTU получен ответ с кодом исключения. Значение кода исключения можно проверить в выходной переменной *ErrorIDEx*.
n) В данных ответа от ведомого устройства Modbus-RTU обнаружен недопустимый код функции, неверный объем принятых данных и т. п.
o) В качестве режима связи по последовательному интерфейсу для указанной дополнительной платы не установлен режим *Ведущее устройство Modbus-RTU (Modbus-RTU master)*.

- Данная команда предусматривает отображение дополнительного кода ошибки в переменной *ErrorIDEx* в случае обнаружения ошибки в ведомом устройстве Modbus-RTU. Дополнительный код ошибки выводится в *ErrorIDEx*, если значение кода ошибки *ErrorID* = *WORD#16#0C10*. Формат отображения: *ErrorIDEx=000000XX*. Значение *XX* см. в описании кодов исключения протокола передачи данных MODBUS.
  Описание кодов исключения протокола передачи данных MODBUS см. в документе *MODBUS Application Protocol Specification (Спецификация протокола прикладного уровня MODBUS)*. Для получения документа *MODBUS Application Protocol Specification* следует обращаться в организацию Modbus Organization, Inc.
  (http://www.modbus.org/)

Пример программы

В данном примере интерфейсный модуль серии NX (NX-CIF210) подключен к интерфейсному модулю EtherCAT (NX-ECC203).
Значение номера модуля, установленное для модуля NX-CIF210: *1*.
В рабочих параметрах модуля для модуля NX-CIF210: задайте параметр **Ch2 Number of Characters to Determine the End (канал 2, число символов для определения конца данных)** равным *35*. Количество символов во время работы принимается за 3,5, поскольку значение параметра Number of Characters to Determine the End (Число символов для определения конца данных) задается в масштабе 1 = 0,1 символа.



Когда переменная *Trigger* принимает значение ИСТИНА, команда программы очищает буфер последовательного порта 2 модуля NX-CIF210, после чего передает команду Modbus-RTU. Изменяется состояние выхода по начальному адресу записи 149 в ведомом устройстве с адресом 1.
Для записи значения в переменную передаются/принимаются команды записи.

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Stage	INT	0	
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	DevicePort	_sDEVICE_PORT		Параметры порта
	NX_SerialBufClear_instance	NX_SerialBufClear		Очистка буфера
	ClearDone	BOOL		
	ClearError	BOOL		
	NX_ModbusRtuWrite_instance	NX_ModbusRtuWrite		
	ModbusSlaveAdr	UINT	UINT#0	Адрес ведомого устройства
	ModbusDone	BOOL		
	ModbusCommandAborted	BOOL		
	ModbusError	BOOL		
	DoModbusTrigger	BOOL		
	ModbusWriteDat	ARRAY[0..5] OF BOOL	[6(ЛОЖЬ)]	
	ModbusWriteCmd	_sSERIAL_MODBUSRTU_WRITE		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	N1_Node_location_information	_sNXUNIT_ID	✉	

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF (Trigger=TRUE) AND (DoModbusTrigger=FALSE) THEN
    DoModbusTrigger := TRUE;

    NX_SerialBufClear_instance(Execute := FALSE,
        DevicePort:=DevicePort);
    NX_ModbusRtuWrite_instance(Execute:= FALSE,
        DevicePort:=DevicePort,
        WriteDat:=ModbusWriteDat);
    Stage := 1; // Инициализация завершена.
END_IF;

IF (DoModbusTrigger=TRUE) THEN
    CASE Stage OF
    1: // Запрос на очистку буфера
        DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
        DevicePort.NxUnit:=N1_Node_location_information;
        DevicePort.PortNo:=2;

        NX_SerialBufClear_instance(Execute := TRUE,
            DevicePort:=DevicePort,
            Done => ClearDone,
            Error => ClearError);

        IF (ClearDone = TRUE) THEN
            Stage := 2; // Очистка буфера нормально завершена.
        ELSIF (ClearError = TRUE) THEN
            Stage := 99; // Очистка буфера завершена с ошибкой.
        END_IF;

    2: // Запрос записи по протоколу Modbus
        ModbusSlaveAdr := 1; // Адрес ведомого устройства
        ModbusWriteCmd.Fun:=_MDB_WRITE_SINGLE_COIL; // Код функции
        ModbusWriteCmd.WriteAdr:=149; // Адрес записи
        ModbusWriteCmd.WriteSize:=1; // Размер записи

        NX_ModbusRtuWrite_instance(Execute:= TRUE,
            DevicePort:=DevicePort,
            SlaveAdr:=ModbusSlaveAdr,
            WriteCmd:=ModbusWriteCmd,
            WriteDat:=ModbusWriteDat,
            Done=>ModbusDone,
            CommandAborted=>ModbusCommandAborted,
            Error=>ModbusError);

        IF (ModbusDone = TRUE) THEN
            Stage := 3; // Команда NX_ModbusRtuWrite завершена нормально.
        ELSIF (ModbusError=TRUE) OR (ModbusCommandAborted=TRUE) THEN
            Stage :=99; // Команда NX_ModbusRtuWrite завершена с ошибкой или по Abo
rt..
```

```
        END_IF;

    3: // Обработка после нормального завершения команды NX_ModbusRtuWrite.
        Trigger := FALSE;
        DoModbusTrigger := FALSE;

    99: // Обработка ошибки
        Trigger := FALSE;
        DoModbusTrigger := FALSE;
    END_CASE;
END_IF;
```

NX_SerialSigCtl

Команда NX_SerialSigCtl включает или выключает сигнал ER или RS последовательного порта интерфейсного модуля серии NX или последовательного порта дополнительной платы.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_SerialSigCtl	Включение/ выключение сигнала управления последовательного порта	FB	NX_SerialSigCtl_instance NX_SerialSigCtl Execute, DevicePort, Kind, Sig, TimeOut Done, Busy, Error, ErrorID	NX_SerialSigCtl_instance(Execute , DevicePort, Kind, Sig, TimeOut, Done, Busy, Error, ErrorID);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
Kind	Команда сигнала		Команда сигнала	_RS_SIG _ER_SIG*1	---	*2
Sig	Команда включения/ выключения		Команда включения/ выключения	Зависит от типа данных.	---	*2
TimeOut	Время ожидания		Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	Зависит от типа данных.	0,1 с	0

*1. _CS_SIG или _DR_SIG использовать нельзя. Если будет указано одно из этих значений, при выполнении команды произойдет ошибка.

*2. Если опустить входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1499.																			
Kind	Сведения о перечислителях перечислимого типа _eSERIAL_SIG см. в разделе *Функция* на стр. 2-1499.																			
Sig	OK																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TimeOut							OK													

Функция

Команда NX_SerialSigCtl включает или выключает сигнал ER или RS последовательного порта интерфейсного модуля серии NX или последовательного порта дополнительной платы.

Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

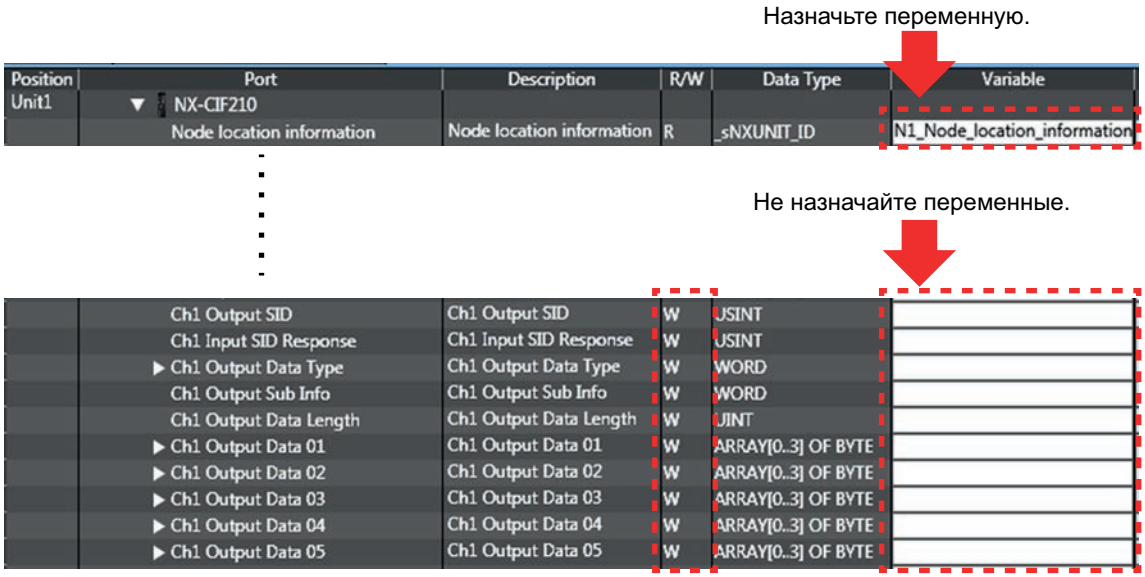
Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.
Задайте значение _DeviceNXUnit для модуля NX или значение _DeviceOptionBoard для дополнительной платы.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

Чтобы указать модуль NX, используйте переменную *NxUnit* для указания устройства.
Переменные *EcatSlave* и *OptBoard* в этом случае не используются.
В переменную *NxUnit* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Чтобы указать дополнительную плату, используйте переменную *OptBoard* для указания устройства.
Переменные *NxUnit* и *EcatSlave* в этом случае не используются.
В переменную *OptBoard* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода, следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2
Для модуля NX укажите порт 1 или порт 2.
Для дополнительной платы укажите порт 1.

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceNXUnit или _DeviceOptionBoard.

Используйте входную переменную *Kind* для выбора сигнала ER или RS.
Когда входная переменная *Sig* = ИСТИНА, сигнал ER или RS включен.
Когда входная переменная *Sig* = ЛОЖЬ, сигнал ER или RS выключен.

Для переменной *Kind* используется перечислимый тип данных _eSERIAL_SIG.
Значения перечислителей перечислимого типа _eSERIAL_SIG приведены в таблице ниже.

Перечислитель	Значение
_RS_SIG	Сигнал RS
_ER_SIG	Сигнал ER
_CS_SIG	Сигнал CS
_DR_SIG	Сигнал DR

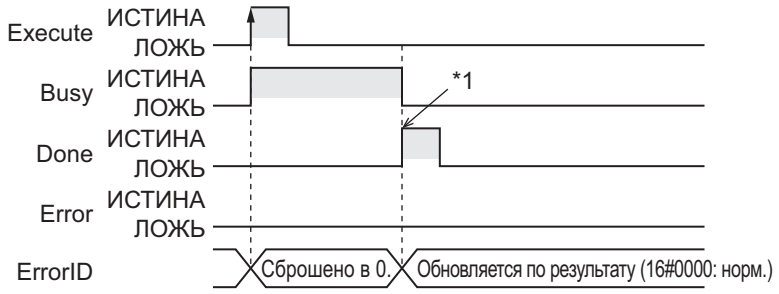
Меры предосторожности для обеспечения надлежащей эксплуатации

Если эта команда будет выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля, произойдет ошибка.

Временные диаграммы

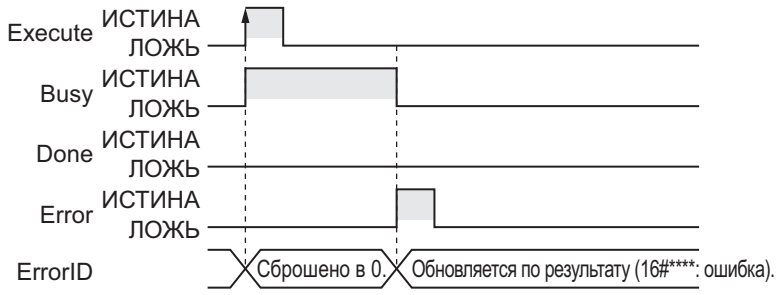
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Управление включением/выключением сигнала завершено.

● Завершение с ошибкой



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_OptBoardSta*1	Состояние дополнительной платы	ARRAY[1..2] of _sOPTBOARD_STA	Хранит данные о состоянии дополнительной платы.
_NXB_UnitIOActiveTbl*2	Состояние активности данных ввода-вывода модуля NX	ARRAY[0..32] OF BOOL*3	• Эти данные о состоянии сообщают, может ли тот или иной модуль NX участвовать в обмене данными ввода-вывода. • Индекс элемента массива соответствует номеру модуля NX. Индексу 0 соответствует ведущее устройство шины NX.

*1. Эту переменную можно использовать только с модулями ЦПУ NX1P2.

*2. Эту переменную можно использовать только с модулями ЦПУ NX102 и NX1P2.
*3. Тип данных для модулей ЦПУ NX1P2: *ARRAY [0..8] OF BOOL*.

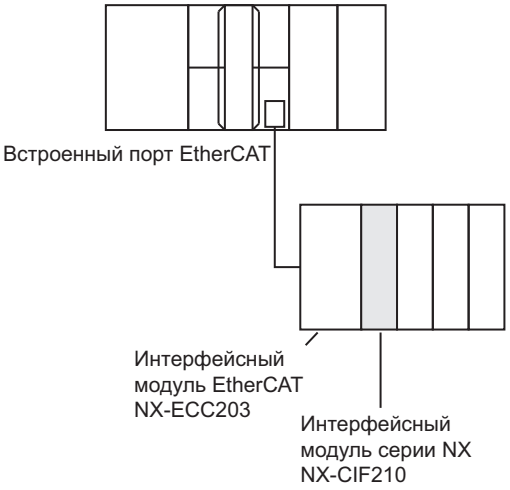
Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- Эта команда не проверяет протокол связи и состояние проводных и кабельных соединений. Перед ее использованием необходимо проверить протокол связи и состояние проводных и кабельных соединений.
- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для переменной *Kind*, *DevicePort.DevicePortType* или *DevicePort.PortNo* задано значение вне допустимого диапазона.
  b) Модуля, дополнительной платы или порта, указанного с помощью *DevicePort*, не существует.
  c) В переменной *DevicePort* указан последовательный порт RS-422A/485.
  d) Если в параметре управления потоком данных для интерфейсного модуля серии NX выбрано значение **RS/CS flow control (Управление потоком с помощью RS/CS)**, а эта команда передает *RS Signal ON (Включить сигнал RS)* или *RS Signal OFF (Выключить сигнал RS)*.
  e) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
  f) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.
  Этой другой командой может быть одна из следующих команд: команда NX_SerialSigRead, команда NX_SerialStatusRead, команда NX_SerialSigCtl, команда NX_SerialBufClear, команда NX_SerialStartMon и команда NX_SerialStopMon.
  g) Истекло время ожидания
  h) Эта команда была выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля.
  i) В качестве режима связи по последовательному интерфейсу для указанной дополнительной платы не установлен режим *Без протокола (No-protocol)* или *Ведущее устройство Modbus-RTU (Modbus-RTU master)*.

Пример программы

В данном примере интерфейсный модуль серии NX (NX-CIF210) подключен к интерфейсному модулю EtherCAT (NX-ECC203).

Значение номера модуля, установленное для модуля NX-CIF210: *1*.



При включении сигнала SetER для беспротокольного удаленного узла, подключенного к последовательному порту 2 модуля NX-CIF210, включается сигнал ER. При включении сигнала ResetER для этого же удаленного узла сигнал ER выключается.

Определения глобальных переменных

● Глобальные переменные

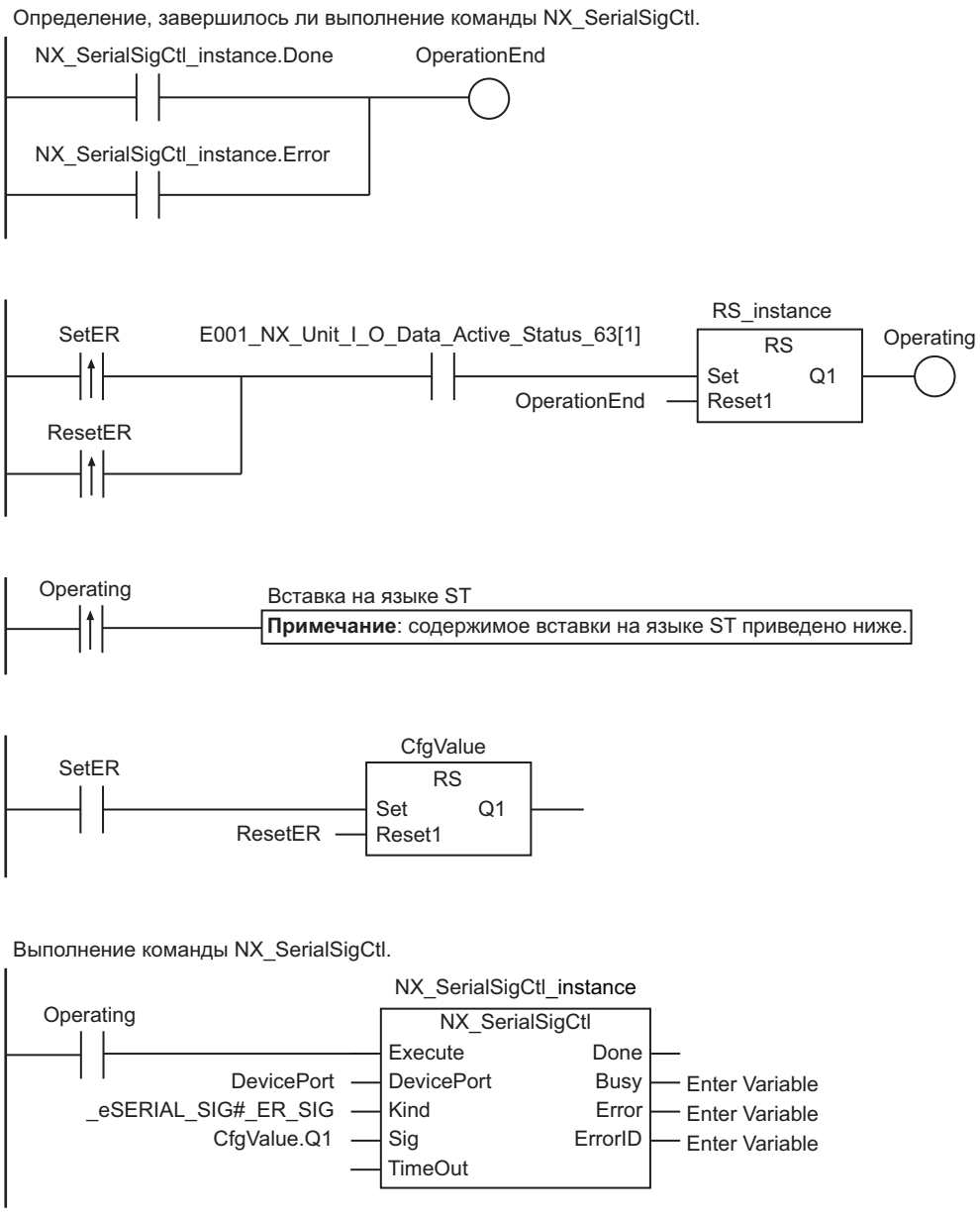
Имя	Тип данных	AT	Комментарий
E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	ECAT://node#1/NX Unit I/O Data Active Status 125	Использование данных ввода-вывода для 63 модулей NX.
N1_Node_location_information	_sNXUNIT_ID	---	Переменная устройства для указания модуля NX-CIF210*1

*1. В Sysmac Studio щелкните правой кнопкой мыши ведомый терминал серии NX, выберите **Display Node Location Port (Отобразить порт расположения узла)** и задайте переменную устройства. Дополнительные сведения см. в: *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperationEnd	BOOL	ЛОЖЬ	Обработка завершена
	SetER	BOOL	ЛОЖЬ	Условие выполнения включения сигнала ER
	ResetER	BOOL	ЛОЖЬ	Условие выполнения выключения сигнала ER
	Operating	BOOL	ЛОЖЬ	Обработка
	DevicePort	_sDEVICE_PORT		Параметры порта
	RS_instance	RS	---	Сохраняется *Operating*
	CfgValue	RS	---	Значение, определяемое переменной *SetER или ResetER*
	NX_SerialSigCtl_instance	NX_SerialSigCtl	---	

Внешние переменные	Переменная	Тип данных	Комментарий
	E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	• Использование данных ввода-вывода для 63 модулей NX. • Если номер соответствующего модуля = *1*, используется E001_NX_Unit_I_O_Data_Active_Status_63[1].
	N1_Node_location_information	_sNXUNIT_ID	Переменная устройства для указания модуля NX-CIF210



● Содержимое вставки на языке ST

```
DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
DevicePort.NxUnit:=N1_Node_location_information;
DevicePort.PortNo:=2;
```

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась
	SetER	BOOL	ЛОЖЬ	Условие выполнения включения сигнала ER
	ResetER	BOOL	ЛОЖЬ	Условие выполнения выключения сигнала ER
	DevicePort	_sDEVICE_PORT		Параметры порта
	CfgValue	RS	---	Значение, определяемое переменной *SetER или ResetER*
	NX_SerialSigCtl_instance	NX_SerialSigCtl	---	

Внешние переменные	Имя	Тип данных	Комментарий
	E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	• Использование данных ввода-вывода для 63 модулей NX. • Если номер соответствующего модуля = *1*, используется E001_NX_Unit_I_O_Data_Active_Status_63[1].
	N1_Node_location_information	_sNXUNIT_ID	Переменная устройства для указания модуля NX-CIF210

```
// Обнаружение SetER или ResetER
IF (NX_SerialSigCtl_instance.Done OR NX_SerialSigCtl_instance.Error) THEN
        OperatingStart:=FALSE;
ELSE_IF
        OperatingStart:=(SetER OR ResetER)
                                AND E001_NX_Unit_I_O_Data_Active_Status_63[1]
                                AND NOT(P_FirstRun);
        DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
        DevicePort.NxUnit:=N1_Node_location_information;
        DevicePort.PortNo:=2;
END_IF;

// Определяется значение сигнала ER.
CfgValue(Set:=SetER, Reset1:=ResetER);

// Выполняется команда NX_SerialSigCtl.
NX_SerialSigCtl_instance(Execute:=OperatingStart,
                         DevicePort:=DevicePort,
                         Kind:=_eSERIAL_SIG#_SIG_ER,
                         Sig:=CfgValue.Q1);
```

NX_SerialSigRead

Команда NX_SerialSigRead считывает состояние сигнала CS или DR последовательного порта дополнительной платы.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_SerialSigRead	Чтение сигнала управления последовательного порта	FB	NX_SerialSigRead_instance NX_SerialSigRead Execute — Done DevicePort — Busy Kind — Error TimeOut — ErrorID Sig	NX_SerialSigRead_instance(Execute, DevicePort, Kind, TimeOut, Done, Busy, Error, ErrorID, Sig);

Меры предосторожности для обеспечения надлежащей эксплуатации

Эту команду можно использовать только для дополнительной платы для модуля ЦПУ NX1P2.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
Kind	Команда сигнала		Команда сигнала	_CS_SIG _DR_SIG*1	---	*2
TimeOut	Время ожидания		Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	Зависит от типа данных.	0,1 с	0
Sig	Сигнал	Выход	Выдает прочитанное состояние сигнала.	Зависит от типа данных.	---	---

*1. _RS_SIG или _ER_SIG использовать нельзя. Если будет указано одно из этих значений, при выполнении команды произойдет ошибка.

*2. Если опустить входной параметр, значение по умолчанию применено не будет. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1507.																			
Kind	Сведения о перечислителях перечислимого типа _eSERIAL_SIG см. в разделе *Функция* на стр. 2-1507.																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TimeOut							OK													
Sig	OK																			

Функция

Команда NX_SerialSigRead производит чтение состояния сигнала CS или DR последовательного порта дополнительной платы.
Если сигнал находится во включенном состоянии, в выходную переменную *Sig* выдается значение ИСТИНА. Если сигнал выключен, *Sig* = ЛОЖЬ.

Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

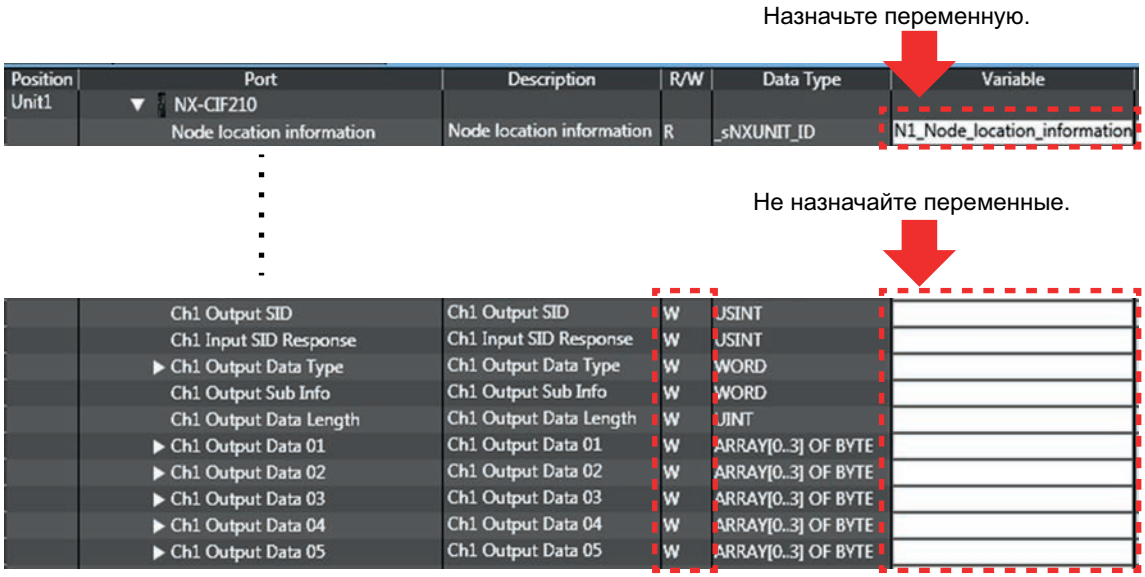
Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.
Для дополнительной платы укажите _DeviceOptionBoard.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

Чтобы указать дополнительную плату, используйте переменную *OptBoard* для указания устройства.
Переменные *NxUnit* и *EcatSlave* в этом случае не используются.

В переменную *OptBoard* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода, следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2
Для дополнительной платы укажите порт 1.

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceOptionBoard.

Используйте входную переменную *Kind* для выбора сигнала CS или DR.

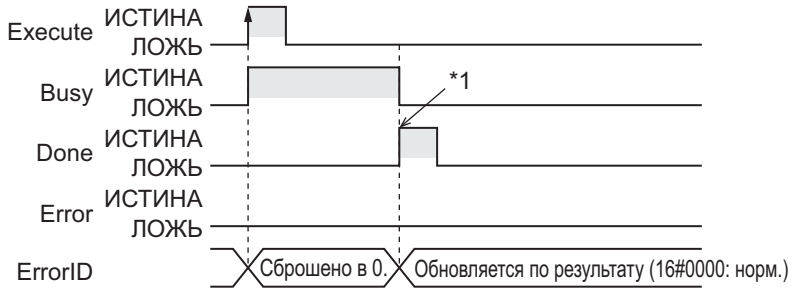
Для переменной *Kind* используется перечислимый тип данных _eSERIAL_SIG.
Значения перечислителей перечислимого типа _eSERIAL_SIG приведены в таблице ниже.

Перечислитель	Значение
_RS_SIG	Сигнал RS
_ER_SIG	Сигнал ER
_CS_SIG	Сигнал CS
_DR_SIG	Сигнал DR

Временные диаграммы

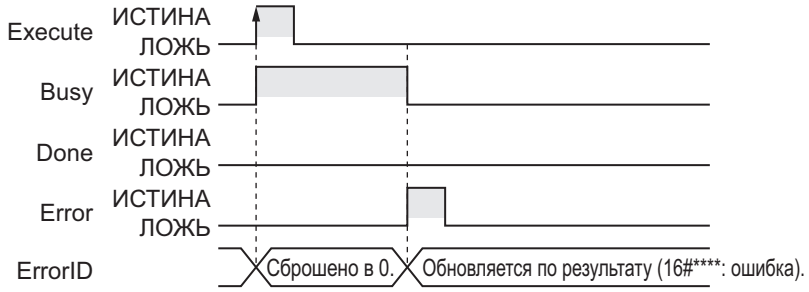
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Считывание сигнала завершено.

● Завершение с ошибкой



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_OptBoardSta	Состояние дополнительной платы	ARRAY[1..2] of _sOPTBOARD_STA	Хранит данные о состоянии дополнительной платы.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- Эта команда не проверяет протокол связи и состояние проводных и кабельных соединений. Перед ее использованием необходимо проверить протокол связи и состояние проводных и кабельных соединений.
- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для

которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.

- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для переменной *Kind*, *DevicePort.DevicePortType* или *DevicePort.PortNo* задано значение вне допустимого диапазона.
  b) Модуля, дополнительной платы или порта, указанного с помощью *DevicePort*, не существует.
  c) В переменной *DevicePort* указан последовательный порт RS-422A/485.
  d) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
  e) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.
  Этой другой командой может быть одна из следующих команд: команда NX_SerialSigCtl, команда NX_SerialSigRead, команда NX_SerialStatusRead, команда NX_SerialBufClear, команда NX_SerialStartMon и команда NX_SerialStopMon.
  f) Истекло время ожидания
  g) Команда была выполнена не для дополнительной платы, а для какого-либо другого устройства.
  h) В качестве режима связи по последовательному интерфейсу для указанной дополнительной платы не установлен режим *Без протокола (No-protocol)* или *Ведущее устройство Modbus-RTU (Modbus-RTU master)*.

Пример программы

См. *Пример программы* на стр. 2-1502 для команды NX_SerialSigCtl.

NX_SerialStatusRead

Команда NX_SerialStatusRead считывает состояние последовательного порта дополнительной платы.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_SerialStatusRead	Чтение состояния последовательного порта	FB	NX_SerialStatusRead_instance NX_SerialStatusRead Execute — Done DevicePort — Busy TimeOut — Error ErrorID PortStatus	NX_SerialStatusRead_instance(Execute, DevicePort, TimeOut, Done, Busy, Error, ErrorID, PortStatus);

Меры предосторожности для обеспечения надлежащей эксплуатации

Эту команду можно использовать только для дополнительной платы для модуля ЦПУ NX1P2.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
TimeOut	Время ожидания		Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	Зависит от типа данных.	0,1 с	0
PortStatus	Состояние порта	Выход	Выдает прочитанное состояние порта.	---	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1511.																			
TimeOut							OK													
PortStatus	Подробные сведения о структуре _sSERIAL_PORT_STATUS см. в разделе *Функция* на стр. 2-1511.																			

Функция

Команда NX_SerialStatusRead считывает состояние последовательного порта дополнительной платы.

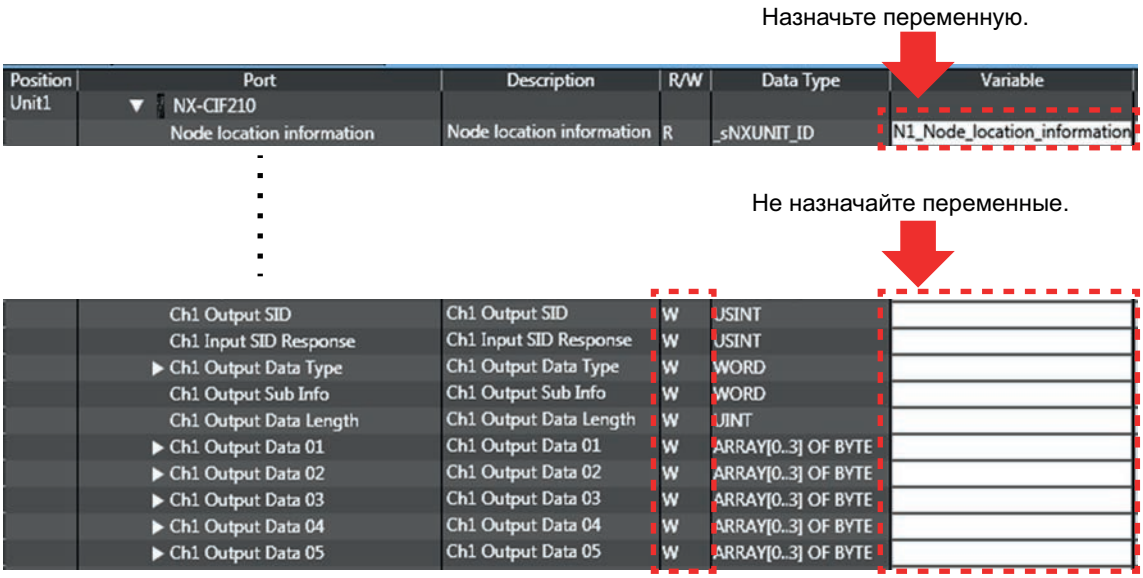
Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.
Для дополнительной платы укажите _DeviceOptionBoard.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

Чтобы указать дополнительную плату, используйте переменную *OptBoard* для указания устройства.
Переменные *NxUnit* и *EcatSlave* в этом случае не используются.
В переменную *OptBoard* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода, следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2
Для дополнительной платы укажите порт 1.

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceOptionBoard.

Для выходной переменной *PortStatus* используется структурный тип данных _sSERIAL_PORT_STATUS.
Описание приведено в таблице ниже.

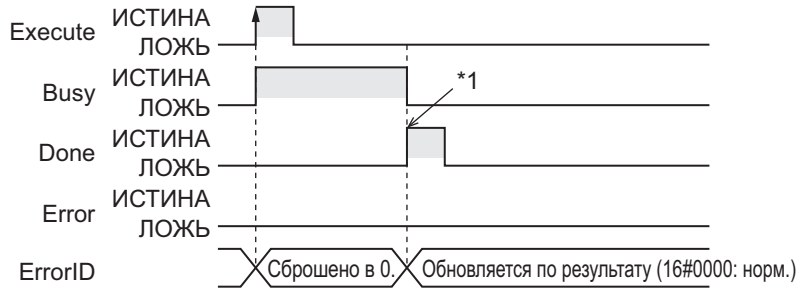
Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
PortStatus	Состояние порта	Выдает прочитанное состояние порта.	_sSERIAL_PORT_STATUS	---	---	---
FullRcvBuf	Данные отброшены из-за переполнения буфера приема	ИСТИНА: данные были отброшены.*1 ЛОЖЬ: данные не были отброшены.	BOOL	Зависит от типа данных.	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---

*1. Данные в буфере приема могут быть неполными.

Временные диаграммы

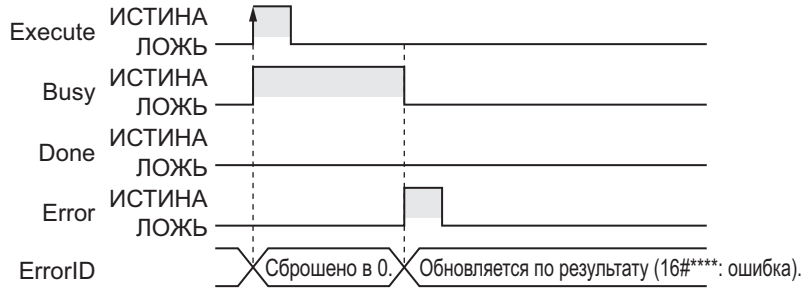
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Чтение состояния порта завершено.

● Завершение с ошибкой



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_OptBoardSta	Состояние дополнительной платы	ARRAY[1..2] of _sOPTBOARD_STA	Хранит данные о состоянии дополнительной платы.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- Эта команда не проверяет протокол связи и состояние проводных и кабельных соединений. Перед ее использованием необходимо проверить протокол связи и состояние проводных и кабельных соединений.
- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для

которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.

- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для переменной *Kind*, *DevicePort.DevicePortType* или *DevicePort.PortNo* задано значение вне допустимого диапазона.
  b) Модуля, дополнительной платы или порта, указанного с помощью *DevicePort*, не существует.
  c) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
  d) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.
  Этой другой командой может быть одна из следующих команд: команда NX_SerialSigCtl, команда NX_SerialSigRead, команда NX_SerialStatusRead, команда NX_SerialBufClear, команда NX_SerialStartMon и команда NX_SerialStopMon.
  e) Истекло время ожидания
  f) Команда была выполнена не для дополнительной платы, а для какого-либо другого устройства.
  g) В качестве режима связи по последовательному интерфейсу для указанной дополнительной платы не установлен режим *Без протокола (No-protocol)* или *Ведущее устройство Modbus-RTU (Modbus-RTU master)*.

Пример программы

См. *Пример программы* на стр. 2-1502 для команды NX_SerialSigCtl.

NX_SerialBufClear

Команда NX_SerialBufClear очищает буфер передачи или приема.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_SerialBuf Clear	Очистка буфе-ра	FB	NX_SerialBufClear_instance NX_SerialBufClear Execute Done DevicePort Busy BufKind Error TimeOut ErrorID	NX_SerialBufClear_instance(Execute, DevicePort, BufKind, TimeOut, Done, Busy, Error, ErrorID);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
BufKind	Тип буфера		Тип буфера (приема или передачи)	_BUF_SENDRCV _BUF_SEND _BUF_RCV	---	_BUFSENDRCV
TimeOut	Время ожидания		Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	Зависит от типа данных.	0,1 с	0

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1517.																			
BufKind	Сведения о перечислителях перечислимого типа _eSERIAL_BUF_KIND см. в разделе *Функция* на стр. 2-1517.																			
TimeOut							ОК													

Функция

Команда NX_SerialBufClear удаляет данные из буфера в соответствии с заданным типом порта и заданным типом буфера. Команда завершается нормально, когда завершается операция очистки.

Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

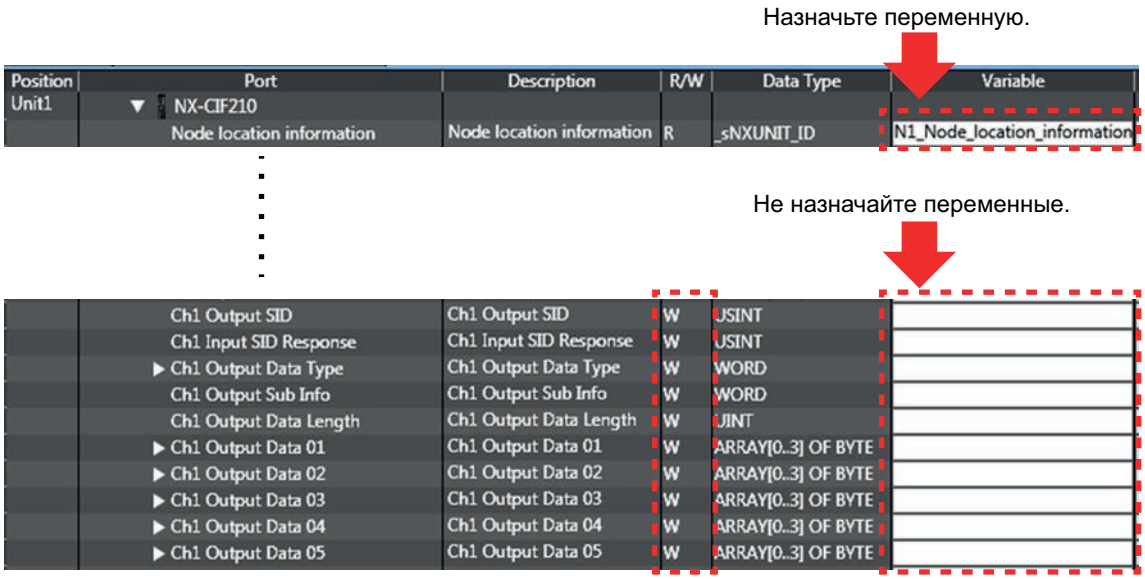
Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.
Задайте значение _DeviceNXUnit для модуля NX или значение _DeviceOptionBoard для дополнительной платы.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

Чтобы указать модуль NX, используйте переменную *NxUnit* для указания устройства.
Переменные *EcatSlave* и *OptBoard* в этом случае не используются.
В переменную *NxUnit* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.
Чтобы указать дополнительную плату, используйте переменную *OptBoard* для указания устройства.
Переменные *NxUnit* и *EcatSlave* в этом случае не используются.
В переменную *OptBoard* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода,

следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2
Для модуля NX укажите порт 1 или порт 2.
Для дополнительной платы укажите порт 1.

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceNXUnit или _DeviceOptionBoard.

Укажите порт с помощью переменной *Port* и укажите тип очищаемого буфера с помощью переменной *BufKind*.

Данные, которые интерфейсный модуль серии NX принял от внешних устройств после очистки буфера приема, не очищаются.

Для переменной *BufKind* используется перечислимый тип данных _eSERIAL_BUF_KIND.
Значения перечислителей перечислимого типа _eSERIAL_BUF_KIND приведены в таблице ниже.

Перечислитель	Значение
_BUF_SENDRCV	Буфер передачи и буфер приема
_BUF_SEND	Буфер передачи
_BUF_RCV	Буфер приема

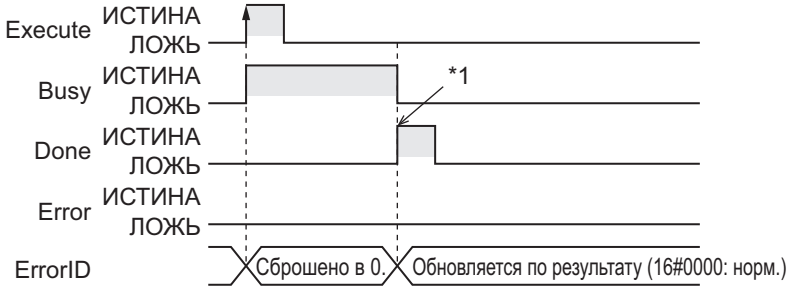
Меры предосторожности для обеспечения надлежащей эксплуатации

Если эта команда будет выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля, произойдет ошибка.

Временные диаграммы

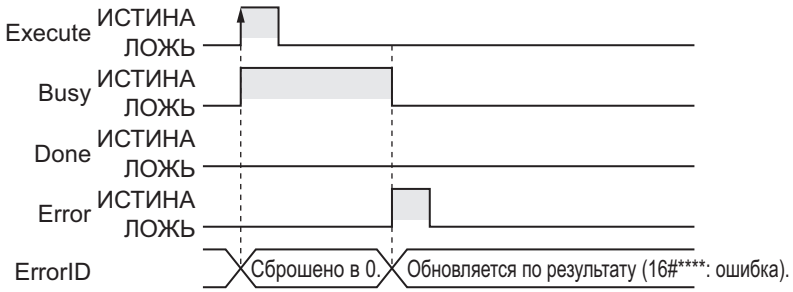
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Операция очистки буфера завершена.

● Завершение с ошибкой



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_PLC_OptBoardSta*1	Состояние дополнительной платы	ARRAY[1..2] of _sOPTBOARD_STA	Хранит данные о состоянии дополнительной платы.
_NXB_UnitIOActiveTbl*2	Состояние активности данных ввода-вывода модуля NX	ARRAY[0..32] OF BOOL*3	• Эти данные о состоянии сообщают, может ли тот или иной модуль NX участвовать в обмене данными ввода-вывода. • Индекс элемента массива соответствует номеру модуля NX. Индексу 0 соответствует ведущее устройство шины NX.

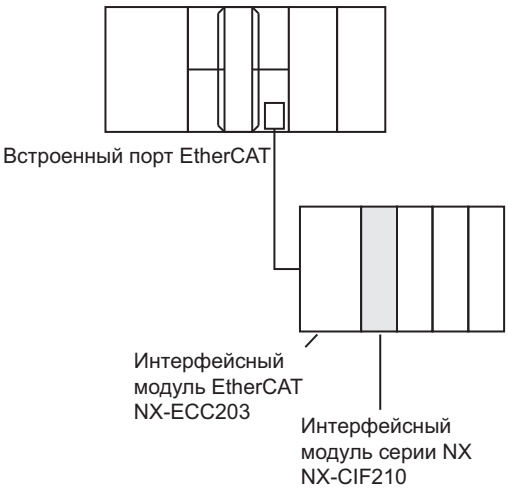
*1. Эту переменную можно использовать только с модулями ЦПУ NX1P2.
*2. Эту переменную можно использовать только с модулями ЦПУ NX102 и NX1P2.
*3. Тип данных для модулей ЦПУ NX1P2: *ARRAY [0..8] OF BOOL*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- Эта команда не проверяет протокол связи и состояние проводных и кабельных соединений. Перед ее использованием необходимо проверить протокол связи и состояние проводных и кабельных соединений.
- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для параметра *BufKind*, *DevicePort.DevicePortType* или *Device- Port.PortNo* указано значение, выходящее за пределы допустимого диапазона.
  b) Модуля, дополнительной платы или порта, указанного с помощью *DevicePort*, не существует.
  c) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
  d) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.
  Этой другой командой может быть одна из следующих команд: команда NX_SerialSend, команда NX_SerialRcv, команда NX_ModbusRtuCmd, команда NX_ModbusRtuRead, команда NX_ModbusRtuWrite, команда NX_SerialSigCtl, NX_SerialSigRead, команда NX_SerialStatusRead, команда NX_SerialBufClear, команда NX_SerialStartMon и команда NX_SerialStopMon.
  e) Истекло время ожидания
  f) Эта команда была выполнена не для интерфейсного модуля серии NX или дополнительной платы, а для какого-либо другого модуля.
  g) В качестве режима связи по последовательному интерфейсу для указанной дополнительной платы не установлен режим *Без протокола (No-protocol)* или *Ведущее устройство Modbus-RTU (Modbus-RTU master)*.

Пример программы

В данном примере интерфейсный модуль серии NX (NX-CIF210) подключен к интерфейсному модулю EtherCAT (NX-ECC203).
Значение номера модуля, установленное для модуля NX-CIF210: *1*.



Эта команда очищает буфер приема последовательного порта 2 модуля NX-CIF210. После завершения операции очистки команда ожидает данные, не содержащие код начала и содержащие код конца данных *CR*.

Определения глобальных переменных

● Глобальные переменные

Имя	Тип данных	AT	Комментарий
E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	ECAT://node#1/NX Unit I/O Data Active Status 125	Использование данных ввода-вывода для 63 модулей NX.
N1_Node_location_information	_sNXUNIT_ID	---	Переменная устройства для указания модуля NX-CIF210*1

*1. В Sysmac Studio щелкните правой кнопкой мыши ведомый терминал серии NX, выберите **Display Node Location Port (Отобразить порт расположения узла)** и задайте переменную устройства. Дополнительные сведения см. в: *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

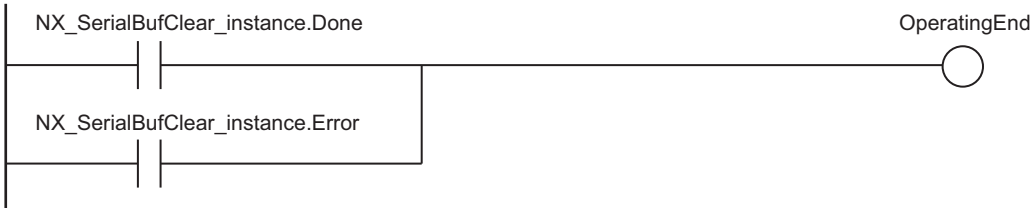
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Операция очистки буфера завершена
	Trigger	BOOL	ЛОЖЬ	Условие выполнения очистки буфера
	Operating	BOOL	ЛОЖЬ	Выполняется операция очистки буфера
	SelectSendBuf	BOOL	ЛОЖЬ	Выбор буфера передачи
	SelectRcvBuf	BOOL	ЛОЖЬ	Выбор буфера приема
	BufKind	_eSERIAL_BUF_KIND	_BUF_SENDRCV	Настройка буфера
	DevicePort	_sDEVICE_PORT		Параметры порта

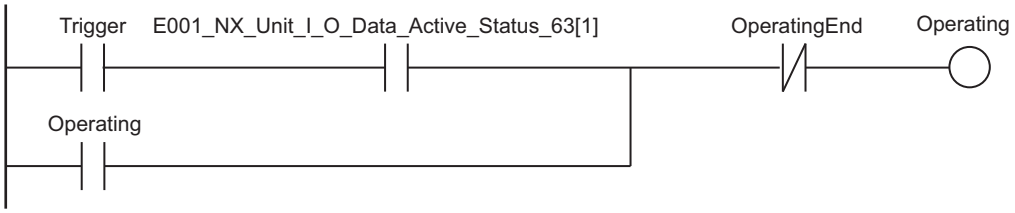
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	NX_SerialBufClear_instance	NX_SerialBufClear	---	
	RcvingEnd	BOOL		Обработка приема завершена
	Rcving	BOOL		Выполняется обработка приема
	RcvCfg	_sSERIAL_CFG		Настройка завершения приема
	StartTrig	_eSERIAL_START	_SERIAL_START_NONE	
	StartCode	ARRAY[0..1] OF BYTE	[2(16#0)]	
	EndTrig	_eSERIAL_END	_SERIAL_END_CODE1	
	EndCode	ARRAY[0..1] OF BYTE	[16#0D, 16#00]	Код конца данных: CR
	RcvSizeCfg	UINT	0	
	Option	_sSERIAL_RCV_OPTION		
	TimeOut	TIME	TIME#0 c	
	LastDatRcv	BOOL	ЛОЖЬ	
	ClearBuf	BOOL	ЛОЖЬ	

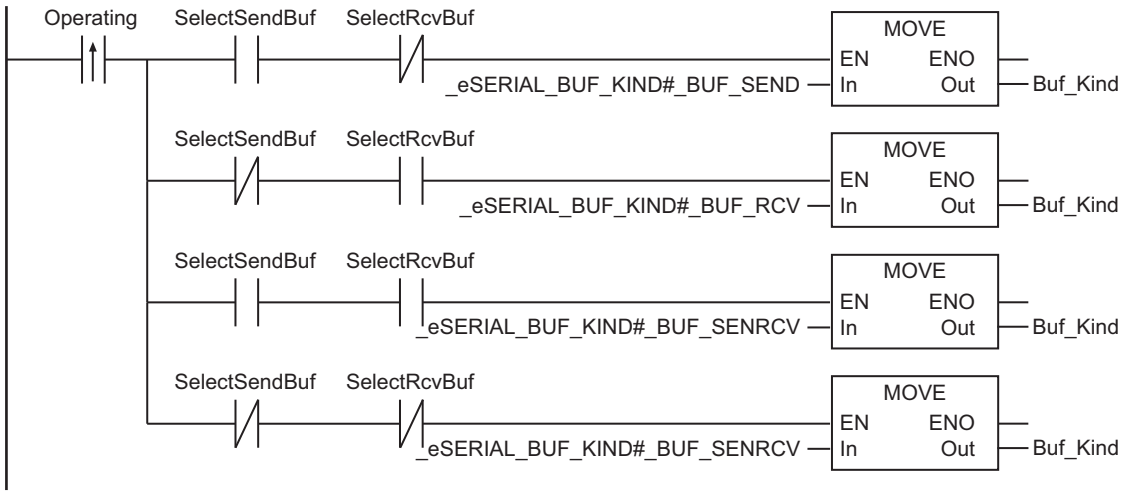
Внешние переменные	Переменная	Тип данных	Комментарий
	E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	• Использование данных ввода-вывода для 63 модулей NX. • Если номер соответствующего модуля = *1*, используется E001_NX_Unit_I_O_Data_Active_Status_63[1].
	N1_Node_location_information	_sNXUNIT_ID	Переменная устройства для указания модуля NX-CIF210

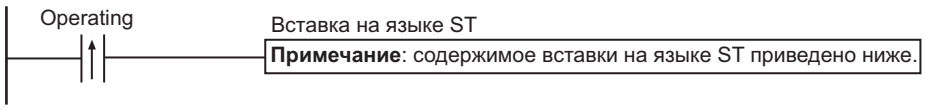
Определение, завершилось ли выполнение команды NX_SerialBufClear.

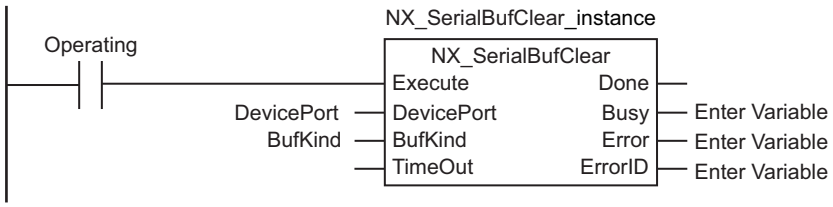


Прием условия выполнения.

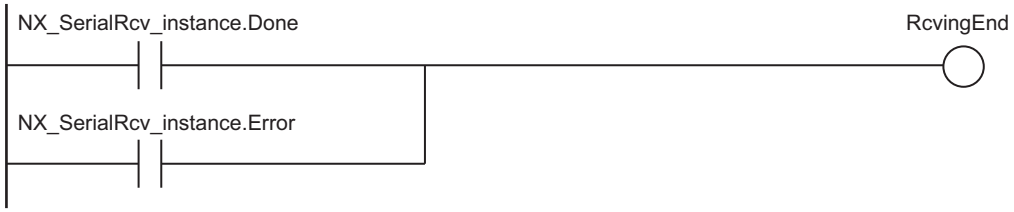


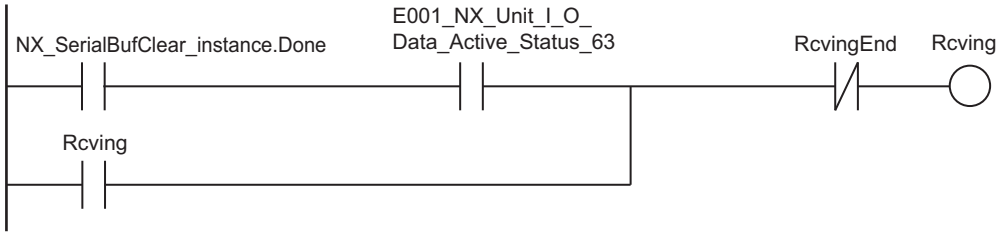




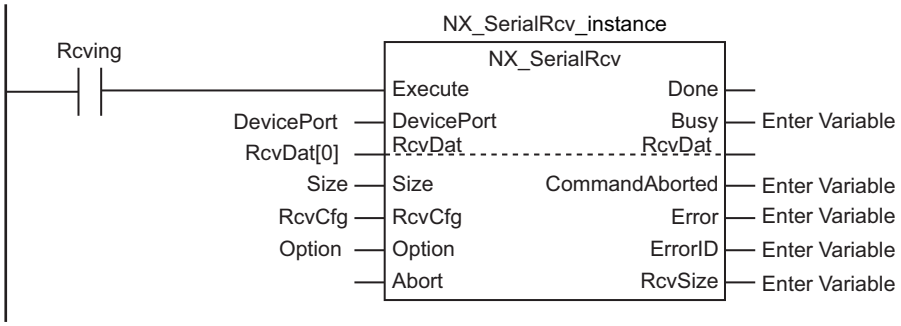


Определение, завершилось ли выполнение команды NX_SerialRcv.





Выполнение команды NX_SerialRcv.



● Содержимое вставки на языке ST

```
DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
DevicePort.NxUnit:=N1_Node_location_information;
DevicePort.PortNo:=2;
```

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Операция очистки буфера завершена
	Trigger	BOOL	ЛОЖЬ	Условие выполнения очистки буфера
	Operating	BOOL	ЛОЖЬ	Выполняется операция очистки буфера
	SelectSendBuf	BOOL	ЛОЖЬ	Выбор буфера передачи
	SelectRcvBuf	BOOL	ЛОЖЬ	Выбор буфера приема
	BufKind	_eSERIAL_BUF_KIND	_BUF_SENDRCV	Настройка буфера
	DevicePort	_sDEVICE_PORT		Параметры порта
	NX_SerialBufClear_instance	NX_SerialBufClear	---	
	RcvingEnd	BOOL		Обработка приема завершена
	Rcving	BOOL		Выполняется обработка приема
	RcvCfg	_sSERIAL_CFG		Настройка завершения приема
	StartTrig	_eSERIAL_START	_SERIAL_START_NONE	
	StartCode	ARRAY[0..1] OF BYTE	[2(16#0)]	
	EndTrig	_eSERIAL_END	_SERIAL_END_CODE1	
	EndCode	ARRAY[0..1] OF BYTE	[16#0D, 16#00]	Код конца данных: CR
	RcvSizeCfg	UINT	0	
	Option	_sSERIAL_RCV_OPTION		

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	TimeOut	TIME	TIME#0 c	
	LastDatRcv	BOOL	ЛОЖЬ	
	ClearBuf	BOOL	ЛОЖЬ	

Внешние переменные	Переменная	Тип данных	Комментарий
	E001_NX_Unit_I_O_Data_Active_Status_63	ARRAY[0..63] OF BOOL	• Использование данных ввода-вывода для 63 модулей NX. • Если номер соответствующего модуля = *1*, используется E001_NX_Unit_I_O_Data_Active_Status_63[1].
	N1_Node_location_information	_sNXUNIT_ID	Переменная устройства для указания модуля NX-CIF210

```
// Настройка условия
RS_instance1(Set:=Trigger AND E001_NX_Unit_I_O_Data_Active_Status_63[1]
             Reset1:=OperatingEnd,
             Q1=>Operating);
R_Trigger_instance(Clk:=Operating);
IF ( (R_Trigger_instance.Q=TRUE) ) THEN
           DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;
           DevicePort.NxUnit:=N1_Node_location_information;
           DevicePort.PortNo:=2;
           IF( (SelectSendBuf=TRUE) THEN
                     IF(SelectRcvBuf=TRUE) THEN
                              BufKind:=_eSERIAL_BUF_KIND#_BUF_SENDRCV;
                     ELSE
                              BufKind:=_eSERIAL_BUF_KIND#_BUF_SEND;
                     END_IF;
           ELSE
                     IF (SelectRcvBuf=TRUE) THEN
                               BufKind:=_eSERIAL_BUF_KIND#_BUF_RCV;
                     ELSE
                               BufKind:=_eSERIAL_BUF_KIND#_BUF_SENDRCV;
                     END_IF
           END_IF;
END_IF;

// Выполнение очистки буфера
NX_SerialBufClear_instance(Execute:=Operating,
                           DevicePort:=DevicePort,
                           BufKind:=BufKind);

//
RS_instane2(Set:=NX_SerialBufClear.Done AND E001_NX_Unit_I_O_Data_Active_Status_63[
```

```
1],
            Reset1:=NX_SerialRcv_instance.Done OR NX_SerialRcv_instance.Error,
            Q1=>Rcving);

//
NX_SerialRcv_instance(Execute:=Rcving,
                      DevicePort:=DevicePort,
                      RcvDat:=RcvDat[0],
                      Size:=Size,
                      RcvCfg:=RcvCfg,
                      Option:=Option);
```

NX_SerialStartMon

Команда NX_SerialStartMon запускает мониторинг линии последовательного интерфейса интерфейсного модуля серии NX.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_SerialStartMon	Запуск мониторинга линии последовательного интерфейса	FB	NX_SerialStartMon_instance NX_SerialStartMon Execute — Done DevicePort — Busy Continuous — Error TimeOut — ErrorID	NX_SerialStartMon_instance(Execute, DevicePort, Continuous, TimeOut, Done, Busy, Error, ErrorID);

Меры предосторожности для обеспечения надлежащей эксплуатации

Эту команду нельзя использовать для дополнительной платы для модуля ЦПУ NX1P2.

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
Continuous	Непрерывный мониторинг		Способ выполнения мониторинга линии последовательного интерфейса ИСТИНА: продолжительный ЛОЖЬ: однократный	Зависит от типа данных.	---	ЛОЖЬ
TimeOut	Время ожидания		Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	Зависит от типа данных.	0,1 с	0

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1528.																			
Continuous	ОК																			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TimeOut							ОК													

Функция

Команда NX_SerialStartMon запускает мониторинг линии последовательного интерфейса интерфейсного модуля серии NX.
Эта команда нормально завершается после того, как запускается мониторинг линии последовательного интерфейса.

Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

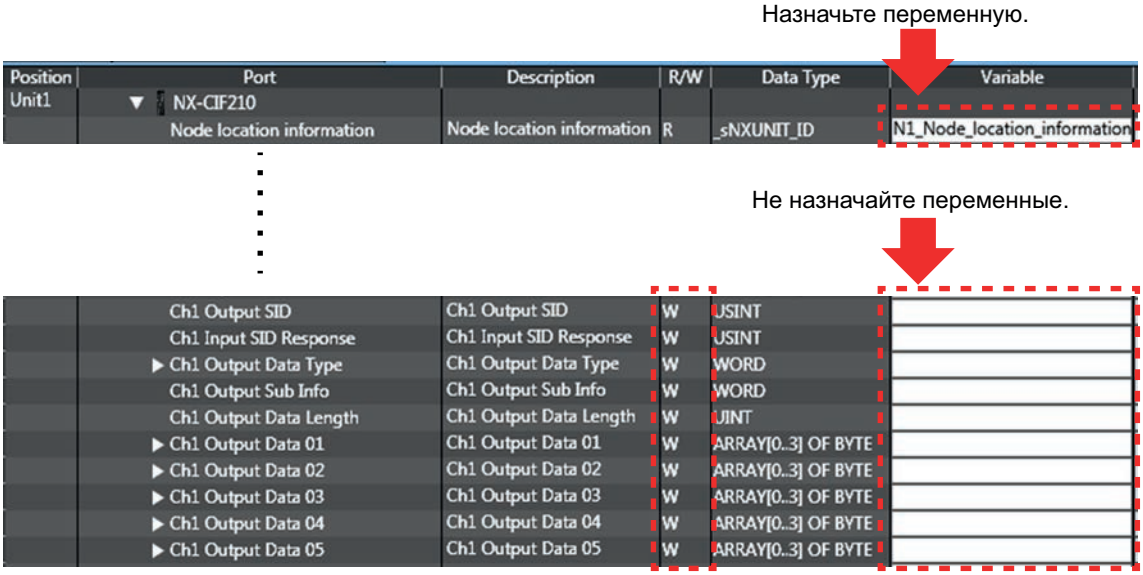
Переменные	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.
Задайте значение _DeviceNXUnit для модуля NX.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

В этой команде для указания устройства используется переменная *NxUnit*. Переменные *EcatSlave* и *OptBoard* не используются.

В переменную *NxUnit* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода, следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceNXUnit.

Если входная переменная *Continuous* = ИСТИНА, значит выбран продолжительный мониторинг. В этом случае мониторинг будет выполняться до тех пор, пока не будет выполнена команда NX_SerialStopMon.
Если входная переменная *Continuous* = ЛОЖЬ, значит выбран однократный мониторинг. В этом случае мониторинг линии последовательного интерфейса будет продолжаться до тех пор, пока не заполнится буфер или не будет выполнена команда NX_SerialStopMon.

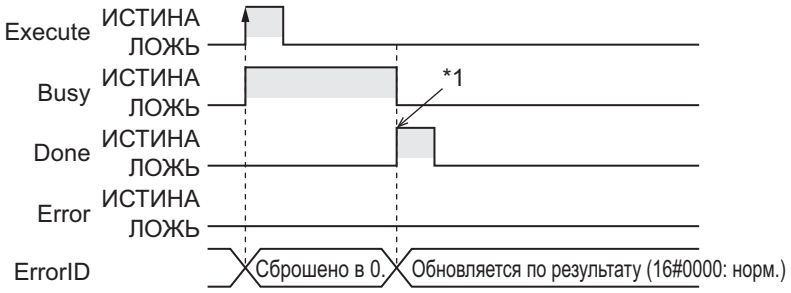
Меры предосторожности для обеспечения надлежащей эксплуатации

Если эта команда будет выполнена не для интерфейсного модуля серии NX, а для какого-либо другого модуля, произойдет ошибка.

Временные диаграммы

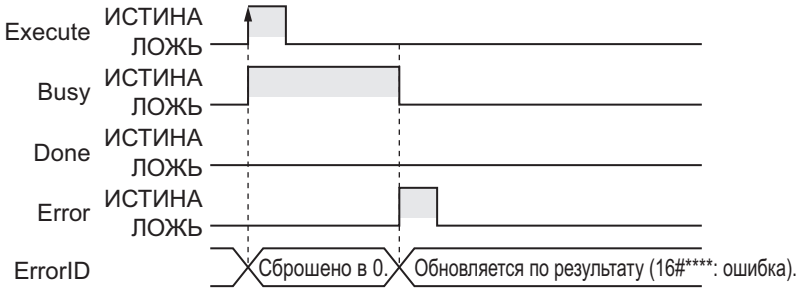
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Начался мониторинг линии последовательного интерфейса.

● Завершение с ошибкой



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_NXB_UnitIOActiveTbl*1	Состояние активности данных ввода-вывода модуля NX	ARRAY[0..32] OF BOOL*2	• Эти данные о состоянии сообщают, может ли тот или иной модуль NX участвовать в обмене данными ввода-вывода. • Индекс элемента массива соответствует номеру модуля NX. Индексу 0 соответствует ведущее устройство шины NX.

*1. Эту переменную можно использовать только с модулями ЦПУ NX102 и NX1P2.
*2. Тип данных для модулей ЦПУ NX1P2: *ARRAY [0..8] OF BOOL*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.

- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для параметра *DevicePort.DevicePortType* или *DevicePort.PortNo* задано значение вне допустимого диапазона.
  b) Модуля, дополнительной платы или порта, указанного с помощью *DevicePort*, не существует.
  c) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
  d) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.
  Этой другой командой может быть одна из следующих команд: команда NX_SerialSigCtl, команда NX_SerialSigRead, команда NX_SerialStatusRead, команда NX_SerialBufClear, команда NX_SerialStartMon и команда NX_SerialStopMon.
  e) Истекло время ожидания
  f) Эта команда была выполнена не для интерфейсного модуля серии NX, а для какого-либо другого модуля.

NX_SerialStopMon

Команда NX_SerialStopMon останавливает мониторинг линии последовательного интерфейса интерфейсного модуля серии NX.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_SerialStop Mon	Остановка мониторинга линии последовательного интерфейса	FB	NX_SerialStopMon_instance NX_SerialStopMon Execute — Done DevicePort — Busy TimeOut — Error ErrorID	NX_SerialStopMon_instance(Execute, DevicePort, TimeOut, Done, Busy, Error, ErrorID);

Меры предосторожности для обеспечения надлежащей эксплуатации

Эту команду нельзя использовать для дополнительной платы для модуля ЦПУ NX1P2.

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.11 или более поздней и Sysmac Studio версии 1.15 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Вход	Объект, представляющий порт устройства	---	---	---
TimeOut	Время ожидания		Время ожидания Если задано значение *0*, время ожидания составляет 2,0 с.	Зависит от типа данных.	0,1 с	0

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DevicePort	Подробные сведения о структуре _sDEVICE_PORT см. в разделе *Функция* на стр. 2-1532.																			
TimeOut							OK													

Функция

Команда NX_SerialStopMon останавливает мониторинг линии последовательного интерфейса интерфейсного модуля серии NX.

Эта команда нормально завершается после того, как останавливается мониторинг линии последовательного интерфейса.

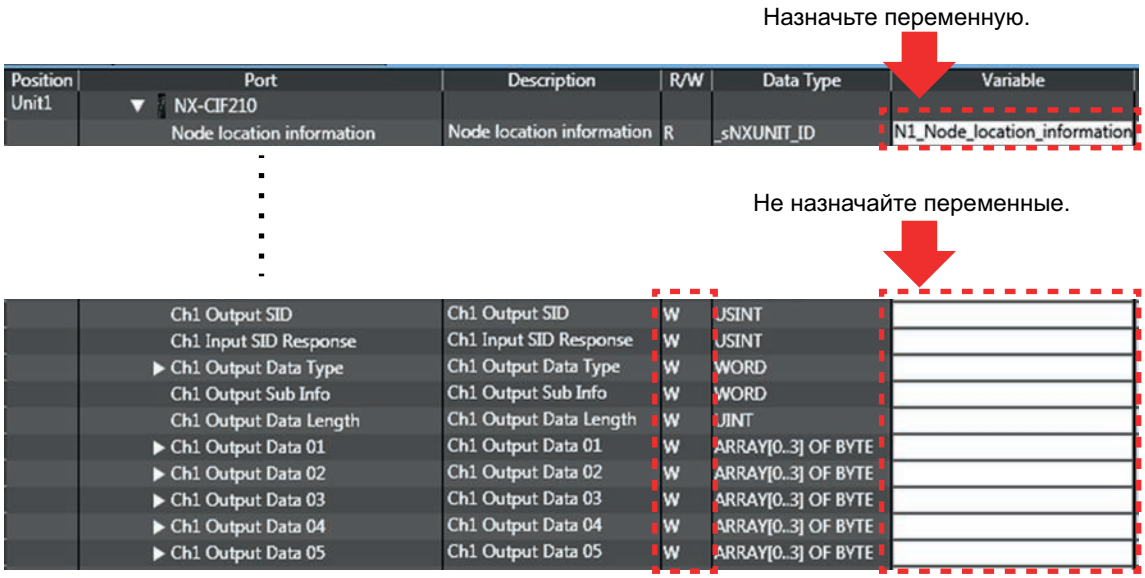
Для входной переменной *DevicePort* используется структурный тип данных _sDEVICE_PORT. Описание приведено в таблице ниже.

Переменные	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
DevicePort	Порт устройства	Объект, представляющий порт устройства	_sDEVICE_PORT	---	---	---
DeviceType	Тип устройства	Тип указываемого устройства	_eDEVICE_TYPE	_DeviceNXUnit _DeviceEcatSlave _DeviceOptionBoard	---	---
NxUnit	Указанный модуль	Модуль NX для управления	_sNXUNIT_ID	---	---	---
EcatSlave	Указанное ведомое устройство	Ведомое устройство EtherCAT для управления	_sECAT_ID	---	---	---
OptBoard	Указанная дополнительная плата	Дополнительная плата для управления	_sOPTBOARD_ID	---	---	---
Reserved	Резерв	Резерв	Резерв	---	---	---
PortNo	Номер порта	Номер порта 1: порт 1 2: порт 2	USINT	Зависит от типа данных.	---	---

Используйте переменную *DeviceType* для указания типа устройства.
Задайте значение _DeviceNXUnit для модуля NX.
Переменная, используемая для указания устройства, определяется указанным типом устройства.

В этой команде для указания устройства используется переменная *NxUnit*. Переменные *EcatSlave* и *OptBoard* не используются.
В переменную *NxUnit* следует передать переменную устройства, которая назначена информации о расположении узла на вкладке I/O Map (Карта входов-выходов) для указываемого устройства.

Если вы используете эту команду, обязательно назначьте переменную устройства информации о расположении узла. Не назначайте переменные устройства каким-либо портам ввода-вывода, следующим за информацией о расположении узла, для которых в столбце R/W («Чтение/запись») указано *W (запись)*.
На рисунке ниже показан пример использования этой команды для порта 1 модуля NX-CIF210.



Дополнительные сведения о назначении переменной устройства для информации о расположении узла см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.

Используйте переменную *PortNo* для указания номера порта.
1: порт 1
2: порт 2

Для параметра *DeviceType* используется перечислимый тип данных _eDEVICE_TYPE.
Значения перечислителей перечислимого типа_eDEVICE_TYPE приведены в таблице ниже:

Перечислитель	Значение
_DeviceNXUnit	Указывается модуль ЦПУ.
_DeviceEcatSlave	Указывается ведомое устройство EtherCAT.
_DeviceOptionBoard	Указывается дополнительная плата.

В этой команде можно указать _DeviceNXUnit.

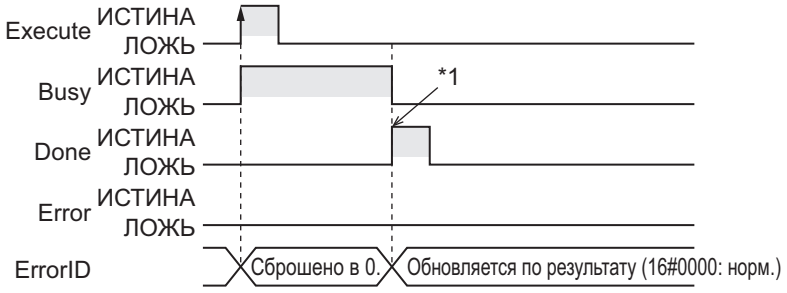
Меры предосторожности для обеспечения надлежащей эксплуатации

Если эта команда будет выполнена не для интерфейсного модуля серии NX, а для какого-либо другого модуля, произойдет ошибка.

Временные диаграммы

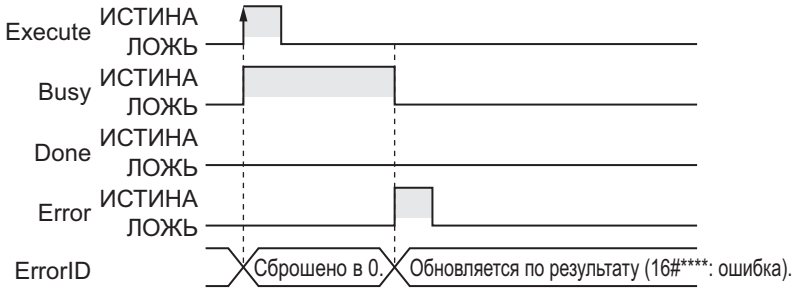
Временные диаграммы работы команды представлены на рисунках ниже.

● Нормальное завершение



*1. Мониторинг линии последовательного интерфейса остановлен.

● Завершение с ошибкой



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_NXB_UnitIOActiveTbl*1	Состояние активности данных ввода-вывода модуля NX	ARRAY[0..32] OF BOOL*2	• Эти данные о состоянии сообщают, может ли тот или иной модуль NX участвовать в обмене данными ввода-вывода. • Индекс элемента массива соответствует номеру модуля NX. Индексу 0 соответствует ведущее устройство шины NX.

*1. Эту переменную можно использовать только с модулями ЦПУ NX102 и NX1P2.
*2. Тип данных для модулей ЦПУ NX1P2: *ARRAY [0..8] OF BOOL*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- При использовании этой команды в событийной задаче возникнет ошибка компиляции. Не применяйте эту команду в событийных задачах.
- При перезапуске интерфейсного модуля серии NX может возникать ошибка *CIF Unit Initialized (Модуль CIF инициализирован)*. При необходимости повторите передачу или прием данных.
- При использовании этой команды не назначайте переменные устройства каким-либо портам ввода-вывода, относящимся к соответствующему интерфейсному модулю серии NX, для которых указано *W (запись)* в столбце R/W («Чтение/запись») на вкладке I/O Map (Карта входов-выходов) в Sysmac Studio.

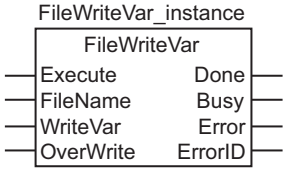
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Для параметра *DevicePort.DevicePortType* или *DevicePort.PortNo* задано значение вне допустимого диапазона.
  b) Модуля, дополнительной платы или порта, указанного с помощью *DevicePort*, не существует.
  c) Было выполнено более 32 следующих команд одновременно: NX_SerialSend, NX_SerialRcv, NX_ModbusRtuCmd, NX_ModbusRtuRead, NX_ModbusRtuWrite, NX_SerialSigCtl, NX_SerialSigRead, NX_SerialStatusRead, NX_SerialBufClear, NX_SerialStartMon и NX_SerialStopMon.
  d) Эта команда выполнена с использованием переменной порта устройства, указанной для другой команды, которая все еще выполняется в данный момент.
  Этой другой командой может быть одна из следующих команд: команда NX_SerialSigCtl, команда NX_SerialSigRead, команда NX_SerialStatusRead, команда NX_SerialBufClear, команда NX_SerialStartMon и команда NX_SerialStopMon.
  e) Истекло время ожидания
  f) Эта команда была выполнена не для интерфейсного модуля серии NX, а для какого-либо другого модуля.

Команды для работы с картой памяти SD

Команда	Имя	Стр.
FileWriteVar	Запись переменной в файл	стр. 2-1538
FileReadVar	Чтение переменной из файла	стр. 2-1544
FileOpen	Открытие файла	стр. 2-1550
FileClose	Закрытие файла	стр. 2-1554
FileSeek	Поиск в файле	стр. 2-1557
FileRead	Чтение из файла	стр. 2-1560
FileWrite	Запись в файл	стр. 2-1569
FileGets	Получить текстовую строку	стр. 2-1577
FilePuts	Поместить текстовую строку	стр. 2-1585
FileCopy	Копирование файла	стр. 2-1594
FileRemove	Удаление файла	стр. 2-1602
FileRename	Изменение имени файла	стр. 2-1607
DirCreate	Создание каталога	стр. 2-1613
DirRemove	Удаление каталога	стр. 2-1616
BackupToMemoryCard	Резервное копирование на карту памяти SD	стр. 2-1620

FileWriteVar

Команда FileWriteVar записывает значение переменной в указанный файл на карте памяти SD. Значение записывается в двоичном формате.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileWriteVar	Запись переменной в файл	FB		FileWriteVar_instance(Execute, FileName, WriteVar, OverWrite, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileName	Имя файла	Вход	Имя файла, в который нужно записать переменную	Макс. 66 байт (65 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	"
WriteVar	Переменная		Записываемая переменная	Зависит от типа данных.		*1
OverWrite	Разрешение перезаписи		ИСТИНА: разрешить перезапись. ЛОЖЬ: запретить перезапись.			ЛОЖЬ

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileName																				OK
WriteVar	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать перечисление, массив, элемент массива, структуру или член структуры.																			
OverWrite	OK																			

Функция

Команда FileWriteVar записывает значение переменной *WriteVar* в файл на карте памяти SD, указанный параметром *FileName*. Значение записывается в двоичном формате.

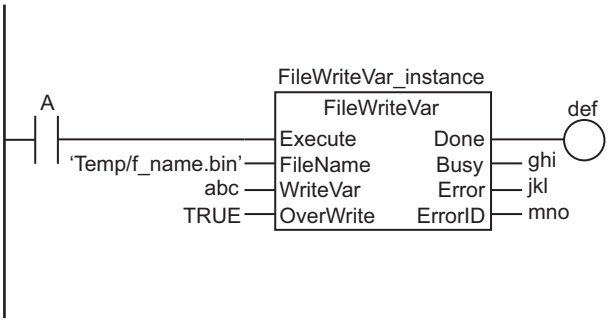
Для *WriteVar* можно указать перечисление, массив, элемент массива, структуру или член структуры.

Если файла с именем *FileName* на карте памяти SD нет, он будет создан.
Переменная *FileName* включает путь. Если указанного каталога на карте памяти SD нет, он будет создан. Каталог, однако, создается, только если не существует каталога самого нижнего уровня в указанном пути.
Если файл с именем *FileName* уже существует на карте памяти SD, выполняется одна из следующих операций в зависимости от значения параметра *OverWrite* (разрешение перезаписи).

Значение *OverWrite*	Операция
ИСТИНА (разрешить перезапись.)	Существующий файл будет перезаписан.
ЛОЖЬ (запретить перезапись.)	Файл не перезаписывается, и возникает ошибка.

Пример программы представлен на рисунке ниже.
Содержимое переменной-массива *abc* записывается в файл с именем «Temp/f_name.bin». Переменная *abc* является переменной-массивом с тремя элементами типа INT.

LD

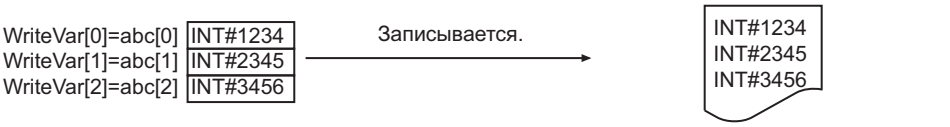


ST

```
FileWriteVar_instance(A, 'Temp/f_name.bin', abc,
                      TRUE, def, ghi, jkl, mno);
```

Команда FileWriteVar записывает значение переменной ***WriteVar*** в файл на карте памяти SD, указанный параметром ***FileName***.
Значение записывается в двоичном формате.

Файл ***FileName*** = «Temp/f_name.bin»



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.

Имя	Значение	Тип данных	Описание
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.
*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

Корневым каталогом в полном имени файла является каталог верхнего уровня карты памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Все данные для записи в команде не сохраняются.
  Согласованность значения может не обеспечиваться, так как значение не синхронизируется с временем выполнения команды и к переменной, передаваемой в *WriteVar*, каждый раз производится обращение.
  Не следует обращаться к целевой переменной во время выполнения команды.
  В случае обращения к переменной во время выполнения команды в файл может быть записано непредусмотренное значение.
- Всегда используйте переменную в качестве входного параметра, передаваемого в *WriteVar*. При передаче константы произойдет ошибка сборки.
- Если параметр *WriteVar* является перечислением, в него нельзя напрямую передавать перечислитель. В случае непосредственной передачи перечислителя произойдет ошибка сборки.
- Если размер указанного файла больше размера *WriteVar*, ошибки не произойдет и будут записаны только данные, соответствующие размеру *WriteVar*. После выполнения этой команды размер указанного файла уменьшается до размера *WriteVar*.
- Данные записываются побайтово. Младшие байты записываются перед старшими байтами (прямой порядок байтов).
- Если *WriteVar* является структурой, между членами структуры могут быть вставлены дополнительные согласующие области в зависимости от состава структуры.

- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- Даже если для записи данных на карту памяти SD используется команда FileWriteVar (Запись переменной в файл) и команда завершается нормально, данные могут быть записаны на карту памяти SD неправильно.
  Если нужно быть уверенным, что данные на карту памяти SD были записаны правильно, в пользовательской программе следует предусмотреть, чтобы записанные данные считывались с помощью команды FileReadVar (Чтение переменной из файла) и сравнивались с исходными данными.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Карта памяти SD защищена от записи.
  c) На карте памяти SD недостаточно свободного места.
  d) Значение *FileName* не является допустимым именем файла.
  e) Файл с именем *FileName* уже существует, и к нему производится доступ.
  f) Файл с именем *FileName* уже существует, и значение *OverWrite* равно ЛОЖЬ.
  g) Файл с именем *FileName* уже существует и защищен от записи.
  h) Длина значения *FileName* превышает максимально допустимое количество байтов в имени файла.
  i) Превышено максимально допустимое количество файлов или каталогов.
  j) Выполняются одновременно пять или больше перечисленных ниже команд для работы с картой памяти SD, не имеющих параметра *FileID*: FileWriteVar, FileReadVar, FileCopy, DirCreate, FileRemove, DirRemove и FileRename.
  k) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

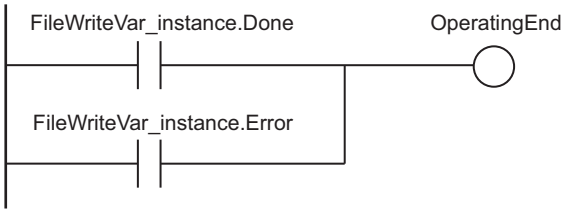
В данном примере все содержимое переменной-массива Var1[] записывается в файл «File1.dat».

Программа на языке LD

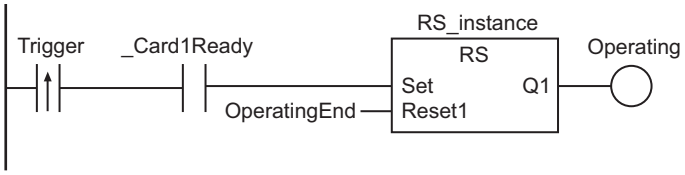
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена.
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	Var1	ARRAY[0..999] OF INT	[1000(0)]	Записываемые данные
	RS_instance	RS		
	FileWriteVar_instance	FileWriteVar		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

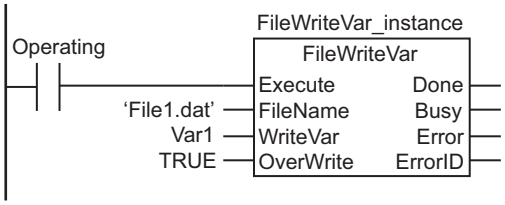
Проверка завершения выполнения команды FileWriteVar.



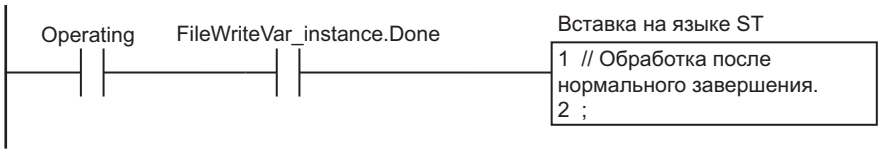
Прием условия выполнения.



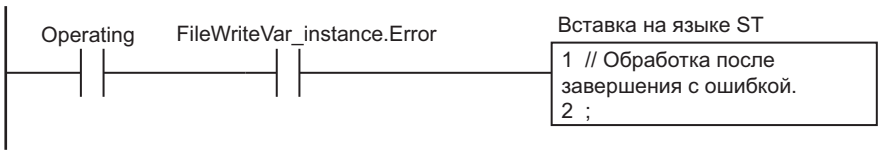
Выполнение команды FileWriteVar.



Обработка после нормального завершения.



Обработка после завершения с ошибкой



Программа на языке ST

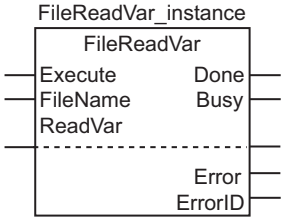
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась.
	Operating	BOOL	ЛОЖЬ	Обработка
	Var1	ARRAY[0..999] OF INT	[1000(0)]	Переменная
	FileWriteVar_instance	FileWriteVar		

Внешние перемен-ные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

```
// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация команды FileWriteVar.
IF (OperatingStart=TRUE) THEN
    FileWriteVar_instance(
        Execute    :=FALSE,
        WriteVar   :=Var1);
    OperatingStart:=FALSE;
END_IF;

// Выполнение команды FileWriteVar.
IF (Operating=TRUE) THEN
    FileWriteVar_instance(
        Execute  :=TRUE,
        FileName :='File1.dat',  // Имя файла
        WriteVar :=Var1,           // Переменная
        OverWrite:=TRUE);          // Разрешить перезапись.

    IF (FileWriteVar_instance.Done=TRUE) THEN
        // Обработка после нормального завершения.
        Operating:=FALSE;
    END_IF;

    IF (FileWriteVar_instance.Error=TRUE) THEN
        // Обработка после завершения с ошибкой.
        Operating:=FALSE;
    END_IF;
END_IF;
```

FileReadVar

Команда FileReadVar считывает содержимое указанного файла на карте памяти SD как двоичные данные и записывает его в переменную.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileReadVar	Чтение переменной из файла	FB		FileReadVar_instance(Execute, FileName, ReadVar, Done, Busy, Error, ErrorID);

Переменные

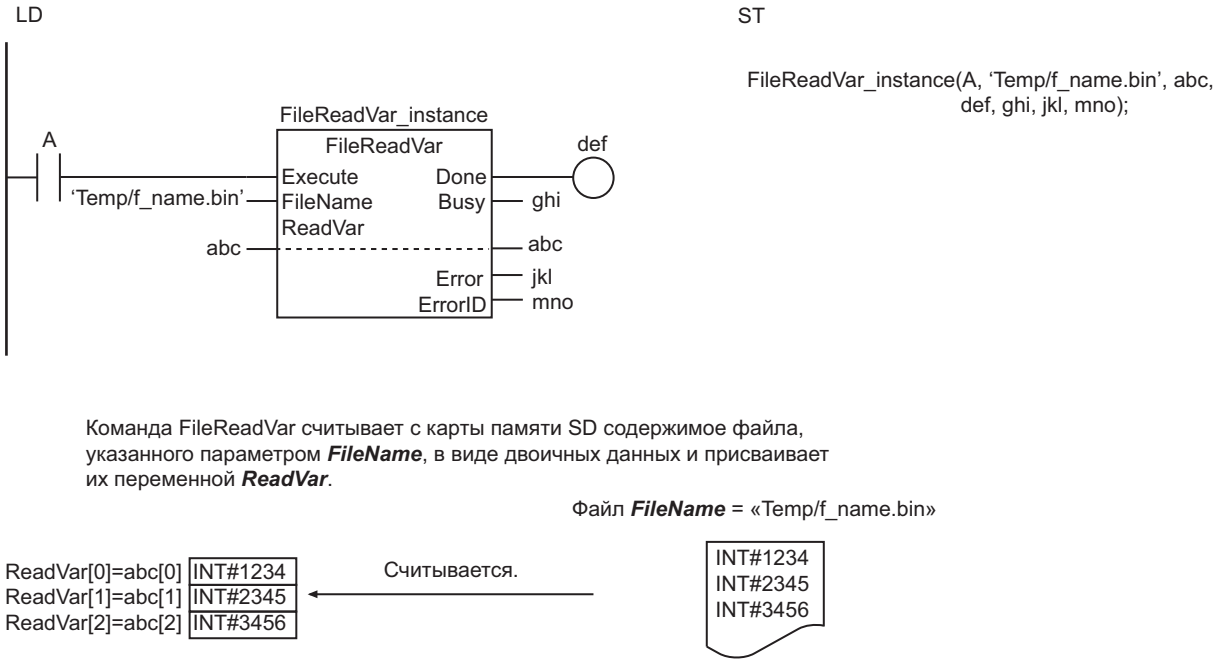
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileName	Имя файла	Вход	Имя файла для чтения	Макс. 66 байт (65 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	"
ReadVar	Записываемая переменная	Вход-выход	Переменная, в которую нужно записать прочитанное значение	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileName																				OK
ReadVar	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также можно указать перечисление, массив, элемент массива, структуру или член структуры.																			

Функция

Команда FileReadVar производит чтение содержимого файла на карте памяти SD, указанного параметром *FileName*, оперируя содержимым как двоичными данными. Прочитанное содержимое присваивается переменной *ReadVar* (записываемая переменная).
Для *ReadVar* можно указать перечисление, массив, элемент массива, структуру или член структуры.

Пример программы представлен на рисунке ниже.
В данном примере содержимое файла с именем «Temp/f_name.bin» считывается и записывается в переменную-массив abc[].
Переменная *abc* является переменной-массивом с тремя элементами типа INT.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.
*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

Корневым каталогом в полном имени файла является каталог верхнего уровня карты памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Все данные для чтения в команде не сохраняются.
  Согласованность значения может не обеспечиваться, так как значение не синхронизируется с временем выполнения команды и переменная, передаваемая в *ReadVar*, каждый раз обновляется.
  Не следует обращаться к целевой переменной во время выполнения команды.
  В случае обращения к переменной во время выполнения команды значение переменной может поменяться на значение, отличающееся от значения, считанного из файла.
- Если размер указанного файла больше размера *ReadVar*, ошибки не произойдет и будут прочитаны только данные, соответствующие размеру *ReadVar*.
- Если размер указанного файла меньше размера *ReadVar*, ошибки не произойдет и будут прочитаны только данные, соответствующие размеру указанного файла. В остальной части *ReadVar* сохранятся значения, которые там находились до выполнения этой команды.
- Данные читаются побайтово. Младшие байты читаются перед старшими байтами (прямой порядок байтов).
- Если *ReadVar* является структурой, между членами структуры могут быть вставлены дополнительные согласующие области в зависимости от состава структуры.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- Для *ReadVar* нельзя указывать переменную устройства. Если будет указана переменная устройства, прочитанное значение не будет присвоено переменной *ReadVar*.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Файла с указанным именем *FileName* не существует.
  c) К файлу с указанным именем *FileName* в данный момент производится доступ.
  d) Значение *FileName* не является допустимым именем файла.
  e) Длина значения *FileName* превышает максимально допустимое количество байтов в имени файла.
  f) Выполняются одновременно пять или больше перечисленных ниже команд для работы с картой памяти SD, не имеющих параметра *FileID*: FileWriteVar, FileReadVar, FileCopy, DirCreate, FileRemove, DirRemove и FileRename.
  g) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

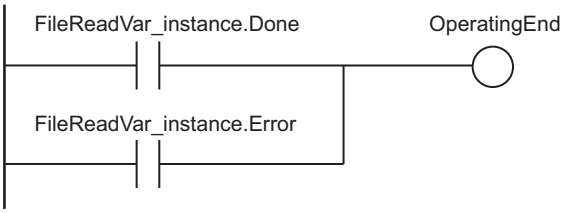
В данном примере производится чтение содержимого файла «File1.dat». Содержимое сохраняется в переменную-массив *Var1*.

Программа на языке LD

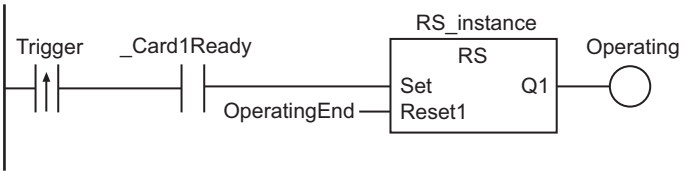
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена.
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	Var1	ARRAY[0..999] OF INT	[1000(0)]	Прочитанные данные
	RS_instance	RS		
	FileReadVar_instance	FileReadVar		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

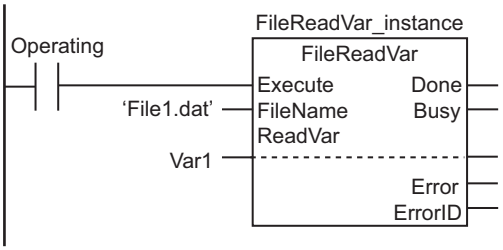
Проверка завершения выполнения команды FileReadVar.



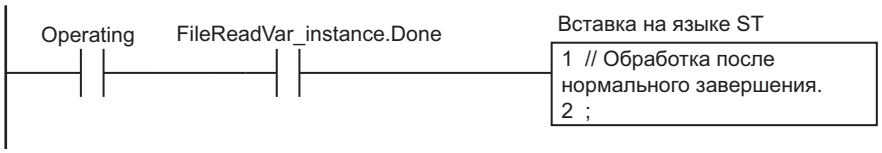
Прием условия выполнения.



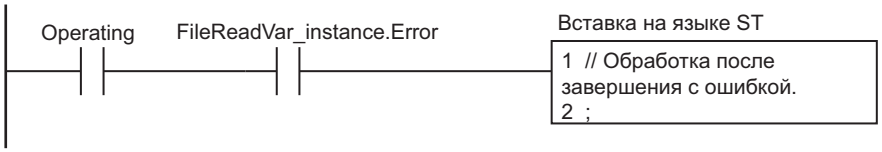
Выполнение команды FileReadVar.



Обработка после нормального завершения.



Обработка после завершения с ошибкой



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась.
	Operating	BOOL	ЛОЖЬ	Обработка
	Var1	ARRAY[0..999] OF INT	[1000(0)]	Переменная для записи
	FileReadVar_instance	FileReadVar		

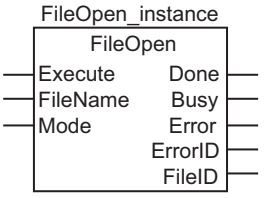
Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

```
// Определение перехода Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация команды FileReadVar.
IF (OperatingStart=TRUE) THEN
    FileReadVar_instance(
        Execute    :=FALSE,
        ReadVar    :=Var1);
    OperatingStart:=FALSE;
END_IF;
```

```
// Выполнение команды FileReadVar.
IF (Operating=TRUE) THEN
    FileReadVar_instance(
        Execute :=TRUE,
        FileName:='File1.dat',    // Имя файла
        ReadVar :=Var1);          // Переменная для записи

    IF (FileReadVar_instance.Done=TRUE) THEN
        // Обработка после нормального завершения.
        Operating:=FALSE;
    END_IF;

    IF (FileReadVar_instance.Error=TRUE) THEN
        // Обработка после завершения с ошибкой.
        Operating:=FALSE;
    END_IF;
END_IF;
```

FileOpen

Команда FileOpen открывает указанный файл на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileOpen	Открытие файла	FB		FileOpen_instance(Execute, FileName, Mode, Done, Busy, Error, ErrorID, FileID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileName	Имя файла	Вход	Имя открываемого файла	Макс. 66 байт (65 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	"
Mode	Режим открытия		Режим открытия файла	*1		_READ_ EXIST
FileID	Идентификатор файла	Выход	Идентификатор файла, который был открыт	Зависит от типа данных.	---	---

*1. _READ_EXIST, _RDWR_EXIST, _WRITE_CREATE, _RDWR_CREATE, _WRITE_APPEND и _RDWR_APPEND

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileName																				OK
Mode	Сведения о перечислителях перечислимого типа _eFOPEN_MODE см. в разделе *Функция* на стр. 2-1550.																			
FileID				OK																

Функция

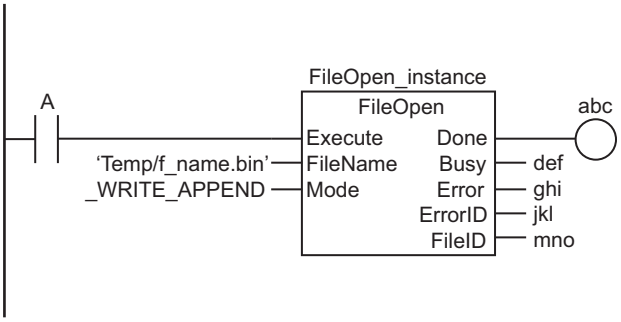
Команда FileOpen открывает файл на карте памяти SD, указанный параметром *FileName*, в режиме, который указан параметром *Mode*.
Результат выводится в переменную *FileID* (идентификатор файла). Значение *FileID* используется для указания файла в других командах, таких как FileRead и FileWrite.

Для параметра *Mode* используется перечислимый тип данных _eFOPEN_MODE. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_READ_EXIST	Это значение используется для открытия текстового файла с целью чтения. Чтение производится с начала файла.
_RDWR_EXIST	Это значение используется для открытия файла с целью чтения и записи. Чтение и запись производятся с начала файла.
_WRITE_CREATE	Это значение используется для открытия файла с целью записи. Если файл уже существует, его содержимое отбрасывается, а размер файла устанавливается равным 0. Если файла не существует, создается новый файл. Запись производится с начала файла. Однако если файл уже существует и он защищен от записи, возникает ошибка и файл не открывается.
_RDWR_CREATE	Это значение используется для открытия файла с целью чтения и записи. Если файл уже существует, его содержимое отбрасывается, а размер файла устанавливается равным 0. Если файла не существует, создается новый файл. Чтение и запись производятся с начала файла.
_WRITE_APPEND	Это значение используется для открытия файла с целью добавления в него данных. Если файла не существует, создается новый файл. Данные добавляются в конец файла. Однако если файл уже существует и он защищен от записи, возникает ошибка и файл не открывается.
_RDWR_APPEND	Это значение используется для открытия файла с целью чтения и добавления в него данных. Если файла не существует, создается новый файл. Чтение производится с начала файла. Данные добавляются в конец файла.

Пример программы представлен на рисунке ниже.
Файл с именем «Temp/f_name.bin» открывается с целью добавления в него данных. Идентификатор файла присваивается переменной *mno*.

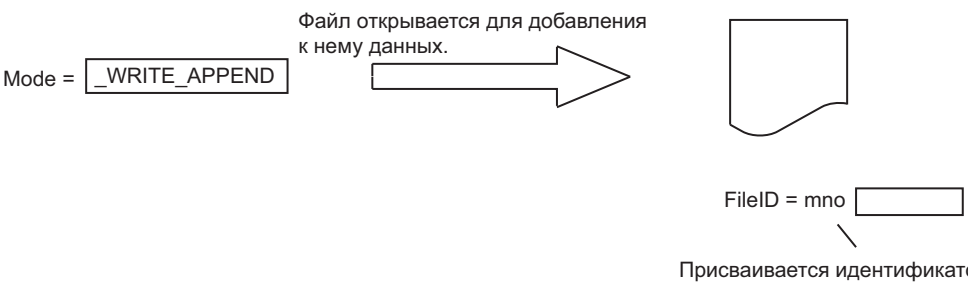
LD



ST

```
FileOpen_instance(A, 'Temp/f_name.bin',
                  _WRITE_APPEND, abc,
                  def, ghi, jkl, mno);
```

Команда FileOpen открывает на карте памяти SD файл, указанный параметром ***FileName***, для добавления к нему данных.
Переменной ***FileID*** присваивается значение идентификатора файла.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

Корневым каталогом в полном имени файла является каталог верхнего уровня карты памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Эта команда должна выполняться перед любой из следующих команд: FileSeek, FileRead, FileWrite, FileGets и FilePuts.
- После завершения использования любого файла, открытого с помощью этой команды, файл необходимо закрыть с помощью команды FileClose.
- После завершения выполнения команды в переменную *FileID* сохраняется значение. В частности, оно сохраняется, когда значение переменной *Done* меняется с ЛОЖЬ на ИСТИНА.

- При переходе модуля ЦПУ в режим «Программирование» и при возникновении ошибки контроллера критического уровня любой открытый файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Если в то время, когда файл открыт, будет отключено питание нажатием кнопки питания карты памяти SD, файл поврежден не будет. Однако файл останется открытым. Файл следует закрыть с помощью команды FileClose.
- Если в то время, когда файл открыт, карта памяти SD будет извлечена из гнезда без нажатия кнопки питания карты памяти SD, содержимое этого файла может быть повреждено. Обязательно выключайте питание карты памяти SD перед ее извлечением.
- Если в то время, когда файл открыт, карта памяти SD будет извлечена из гнезда без нажатия кнопки питания карты памяти SD, файл останется открытым. Файл следует закрыть с помощью команды FileClose.
- Если в то время, когда файл открыт, будет отключено питание или карта памяти SD будет извлечена из гнезда, файл останется открытым, однако операции чтения и записи для этого файла будут невозможны, даже если карта памяти SD вновь будет вставлена. Чтобы выполнить чтение из этого файла или запись в него, файл следует закрыть и вновь открыть.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Карта памяти SD защищена от записи.
  c) Значение *Mode* = _READ_EXIST или _RDWR_EXIST, а файла с указанным именем *FileName* не существует.
  d) Значение *Mode* находится за пределами допустимого диапазона.
  e) К файлу с указанным именем *FileName* в данный момент производится доступ.
  f) Значение *FileName* не является допустимым именем файла.
  g) Файл с указанным именем *FileName* защищен от записи.
  h) Длина значения *FileName* превышает максимально допустимое количество байтов в имени файла.
  i) Предпринята попытка открыть более пяти файлов одновременно.
  j) Превышено максимально допустимое количество файлов или каталогов.
  k) Для модуля ЦПУ версии 1.10 или более поздней: при попытке открыть уже открытый файл возникает ошибка *File Already in Use (Файл уже используется)* и в выходную переменную *FileID* сохраняется идентификатор открытого файла. При возникновении любой другой ошибки выходная переменная *FileID* не изменяется.
  Для модуля ЦПУ версии 1.09 или более ранней: при возникновении ошибки в выходную переменную *FileID* сохраняется значение *0*.
  l) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

См. *Пример программы* на стр. 2-1563 для команды FileRead, *Пример программы* на стр. 2-1572 для команды FileWrite, *Пример программы* на стр. 2-1580 для команды FileGets и *Пример программы* на стр. 2-1587 для команды FilePuts.

FileClose

Команда FileClose закрывает указанный файл на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileClose	Закрытие файла	FB	FileClose_instance FileClose Execute Done FileID Busy Error ErrorID	FileClose_instance(Execute, FileID, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileID	Идентификатор файла	Вход	Идентификатор закрываемого файла	Зависит от типа данных.	---	0

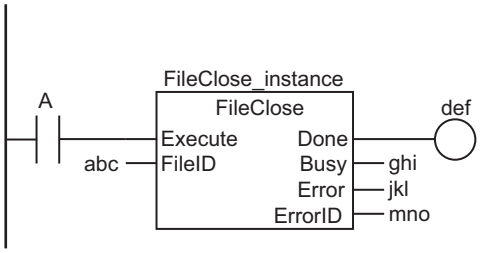
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileID				OK																

Функция

Команда FileClose закрывает файл на карте памяти SD, указанный параметром *FileID*.

Пример программы представлен на рисунке ниже. В данном случае закрывается файл, идентификатор которого равен значению переменной *abc*.

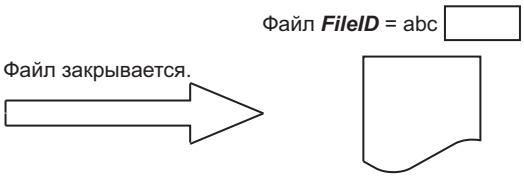
LD



ST

```
FileClose_instance(A, abc, def, ghi, jkl, mno);
```

Команда FileClose закрывает файл на карте памяти SD, указанный параметром ***FileID***.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

Открытие файла с помощью команды FileOpen требуется для следующих команд: FileSeek, FileRead, FileWrite, FileGets и FilePuts.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.

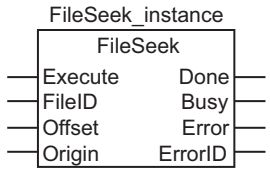
- Для получения значения *FileID* необходимо предварительно использовать команду FileOpen.
- После завершения работы с любым файлом, открытым с помощью команды FileOpen, необходимо использовать эту команду для закрытия файла.
- При переходе модуля ЦПУ в режим «Программирование» и при возникновении ошибки контроллера критического уровня любой открытый файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Если в то время, когда файл открыт, будет отключено питание нажатием кнопки питания карты памяти SD, файл поврежден не будет. Однако файл останется открытым. Файл следует закрыть с помощью команды FileClose.
- Если в то время, когда файл открыт, карта памяти SD будет извлечена из гнезда без нажатия кнопки питания карты памяти SD, содержимое этого файла может быть повреждено. Обязательно выключайте питание карты памяти SD перед ее извлечением.
- Если в то время, когда файл открыт, карта памяти SD будет извлечена из гнезда без нажатия кнопки питания карты памяти SD, файл останется открытым. Файл следует закрыть с помощью команды FileClose.
- Если в то время, когда файл открыт, будет отключено питание или карта памяти SD будет извлечена из гнезда, файл останется открытым, однако операции чтения и записи для этого файла будут невозможны, даже если карта памяти SD вновь будет вставлена. Чтобы выполнить чтение из этого файла или запись в него, файл следует закрыть и вновь открыть.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Файла с указанным идентификатором *FileID* не существует.
  c) Файл с указанным идентификатором *FileID* уже закрыт.
  d) К файлу с указанным идентификатором *FileID* в данный момент производится доступ.
  e) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

См. *Пример программы* на стр. 2-1563 для команды FileRead, *Пример программы* на стр. 2-1572 для команды FileWrite, *Пример программы* на стр. 2-1580 для команды FileGets и *Пример программы* на стр. 2-1587 для команды FilePuts.

FileSeek

Команда FileSeek устанавливает указатель позиции на требуемую позицию в указанном файле на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileSeek	Поиск в файле	FB		FileSeek_instance(Execute, FileID, Offset, Origin, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileID	Идентификатор файла	Вход	Идентификатор файла, в котором нужно установить указатель позиции	Зависит от типа данных.	---	0
Offset	Смещение		Смещение относительно *Origin*		Байты	
Origin	Исходная позиция		Исходная позиция для установки указателя позиции в файле	_SEEK_SET, _SEEK_CUR или _SEEK_END	---	_SEEK_SET

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileID				OK																
Offset												OK								
Origin	Сведения о перечислителях перечислимого типа _eFSEEK_ORIGIN см. в разделе *Функция* на стр. 2-1557.																			

Функция

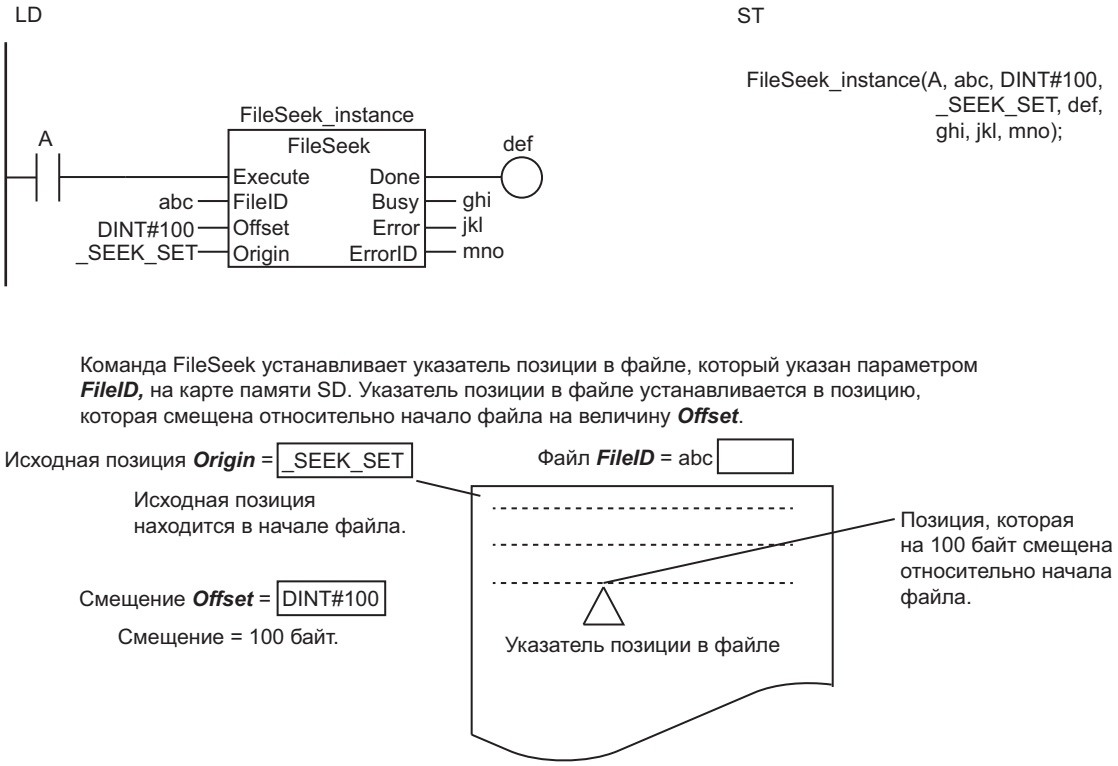
Команда FileSeek устанавливает указатель позиции на требуемую позицию в файле на карте памяти SD, который указан параметром *FileID*.
Указатель позиции задает позицию в файле, откуда должно начинаться чтение или запись данных при выполнении таких команд, как FileRead или FileWrite.
Например, если нужно прочитать данные с начала файла, необходимо установить указатель позиции на начало файла с помощью команды FileSeek, а затем выполнить команду FileRead.

Указатель позиции в файле задается как смещение (*Offset*) относительно исходной позиции (*Origin*).

Для параметра *Origin* используется перечислимый тип данных _eFSEEK_ORIGIN. Значения перечислителей приведены в таблице ниже.

Перечислитель	Значение
_SEEK_SET	Начало файла
_SEEK_CUR	Положение текущего указателя позиции в файле
_SEEK_END	Конец файла

Пример программы представлен на рисунке ниже. Указатель позиции в файле устанавливается в позицию, которая смещена относительно начала файла на 100 байт.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

Имя	Значение	Тип данных	Описание
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.
*2. Подразумевается доступ к карте памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

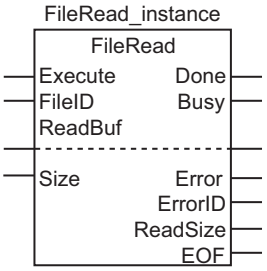
- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Перед выполнением этой команды необходимо использовать команду FileOpen для получения значения *FileID*.
- Если при выполнении команды FileOpen для параметра *Mode* указывается значение **_WRITE_APPEND** или **_RDWR_APPEND** для добавления данных в файл, данные всегда добавляются в конец файла.
  Если при выполнении команды FileOpen для параметра *Mode* указывается значение **_RDWR_APPEND**, указатель позиции в файле, установленный командой FileSeek, используется только для чтения данных.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Значение *Origin* находится за пределами допустимого диапазона.
  c) Позиция, заданная параметрами *Origin* и *Offset*, превышает размер файла.
  d) Файла с указанным идентификатором *FileID* не существует.
  e) К файлу с указанным идентификатором *FileID* в данный момент производится доступ.
  f) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

См. *Пример программы* на стр. 2-1563 для команды FileRead и *Пример программы* на стр. 2-1572 для команды FileWrite.

FileRead

Команда FileRead считывает данные из указанного файла на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileRead	Чтение из файла	FB		FileRead_instance(Execute, FileID, ReadBuf, Size, Done, Busy, Error, ErrorID, ReadSize, EOF);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileID	Идентификатор файла	Вход	Идентификатор читаемого файла	Зависит от типа данных.	---	0
Size	Количество элементов для чтения		Количество элементов для чтения			1
ReadBuf[] (массив)	Буфер чтения	Вход-выход	Буфер, в который нужно записать прочитанные данные	Зависит от типа данных.	---	---
ReadSize	Количество прочитанных элементов	Выход	Количество фактически прочитанных элементов	Зависит от типа данных.	---	---
EOF	Конец файла		Показывает, достигнут ли конец файла. ИСТИНА: достигнут. ЛОЖЬ: не достигнут.			

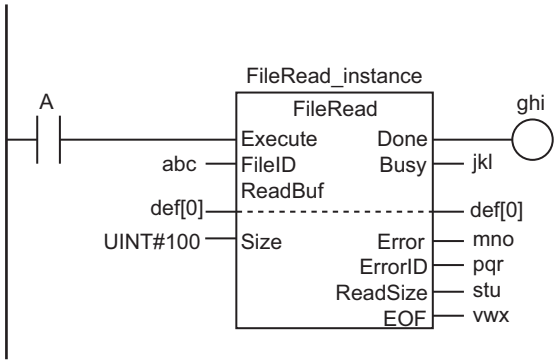
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileID				OK																
Size							OK													
ReadBuf[] (массив)	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также допускается указывать массивы перечислений или структур.																			
ReadSize							OK													
EOF	OK																			

Функция

Команда FileRead считывает данные в указанной позиции файла на карте памяти SD, указанного параметром *FileID*. Прочитанные данные сохраняются в буфер чтения ReadBuf[].
Требуемая позиция чтения определяется указателем позиции, который устанавливается заранее с помощью команды FileSeek.
Объем прочитанных данных в *Size* раз превышает размер, соответствующий типу данных переменной ReadBuf[]. Другими словами, он равен объему *Size* элементов массива ReadBuf[].
Для параметра ReadBuf[] можно указать массив перечислений или структур.
В переменную *ReadSize* записывается фактическое количество прочитанных элементов. Как правило, переменные *Size* и *ReadSize* будут содержать одинаковые значения. Если количество данных после указанной позиции и до конца файла меньше значения *Size*, ошибки не произойдет, а в ReadBuf[] будут сохранены имеющиеся данные до конца файла. В этом случае значение *ReadSize* будет меньше значения *Size*.
Если из файла прочитаны данные до конца файла, значение переменной *EOF* меняется на ИСТИНА. В противном случае в *EOF* остается значение ЛОЖЬ.

Пример программы представлен на рисунке ниже. Если буфер чтения def[] является массивом типа BYTE, из файла для чтения считывается 100 байтов данных.

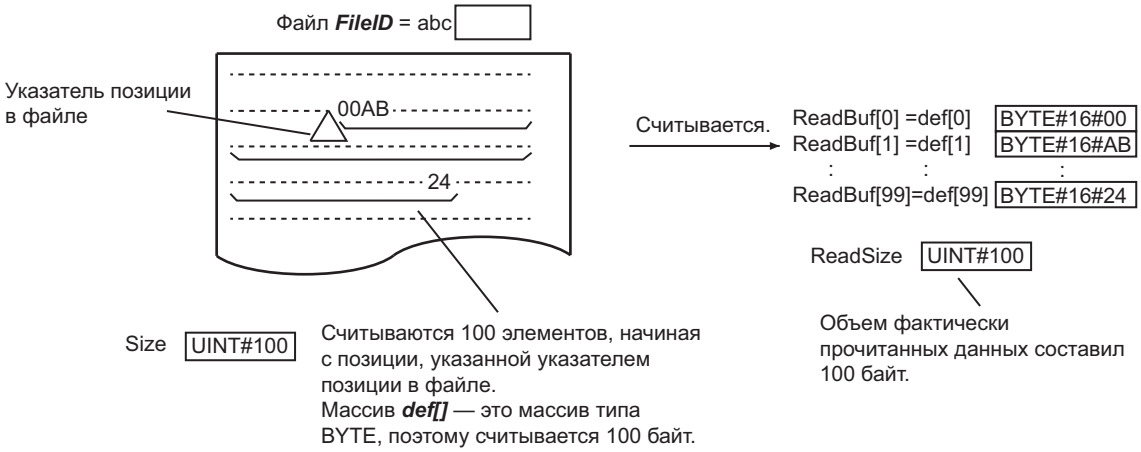
LD



ST

```
FileRead_instance(A, abc, def[0], UINT#100,
                  ghi, jkl, mno, pqr, stu, vwx);
```

Команда FileRead считывает ***Size*** элементов в указанной позиции файла на карте памяти SD, указанного параметром ***FileID***. Прочитанные данные она сохраняет в буфер чтения ***ReadBuf[]***. Фактический размер прочитанных данных выводится в ***ReadSize***.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

*2. Подразумевается доступ к карте памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Все данные для чтения в команде не сохраняются.
  Согласованность значения может не обеспечиваться, так как значение не синхронизируется с временем выполнения команды и переменная, передаваемая в ReadBuf[], каждый раз обновляется.
  Не следует обращаться к целевой переменной во время выполнения команды.
  Если обратиться к переменной во время выполнения команды, в переменную может быть сохранено значение, отличающееся от значения, считанного из файла.
- Если прочитаны данные до конца файла и объем данных не делится без остатка на размер типа данных ReadBuf[], остаток данных, которого недостаточно для размера данных ReadBuf[], отбрасывается. Указатель позиции в файле перемещается в конец файла, и значение *EOF* меняется на ИСТИНА.
- В остальных элементах массива ReadBuf[] (когда он содержит больше элементов, чем указано параметром *Size*) (т. е. в элементах, в которые не производится запись при чтении данных), сохраняются значения, содержавшиеся в них до выполнения этой команды.

- Перед выполнением этой команды необходимо использовать команду FileOpen для получения значения *FileID*.
- Данные читаются побайтово. Младшие байты читаются перед старшими байтами (прямой порядок байтов).
- После завершения выполнения команды в переменную *EOF* сохраняется значение. В частности, оно сохраняется, когда значение переменной *Done* меняется с ЛОЖЬ на ИСТИНА.
- Если ReadBuf[] является массивом структур, между членами структур могут вставляться дополнительные согласующие области в зависимости от состава структуры.
- Если во время выполнения этой команды модуль ЦПУ перейдет в режим «Программирование» или возникнет ошибка контроллера критического уровня, файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- Для ReadBuf[] нельзя указывать переменную устройства. Если будет указана переменная устройства, прочитанные данные не будут присвоены переменной ReadBuf[].
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Количество элементов массива в ReadBuf[] меньше значения *Size*.
  c) Файла с указанным идентификатором *FileID* не существует.
  d) К файлу с указанным идентификатором *FileID* в данный момент производится доступ.
  e) Файл, указанный идентификатором *FileID*, не был открыт в режиме чтения.
  f) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

В этом примере читаются четыре байта данных, начиная со второго байта от начала файла с именем «ABC.bin». Данные записываются в переменную-массив InDat[] типа BYTE.
Обработка выполняется в следующем порядке:

1 Используется команда FileOpen для открытия файла «ABC.bin».

2 С помощью команды FileSeek указатель позиции в файле устанавливается на второй байт от начала файла.

3 Используется команда FileRead для чтения из файла четырех байтов данных в указанной позиции и сохранения их в переменную-массив InDat[].

4 Используется команда FileClose для закрытия файла «ABC.bin».

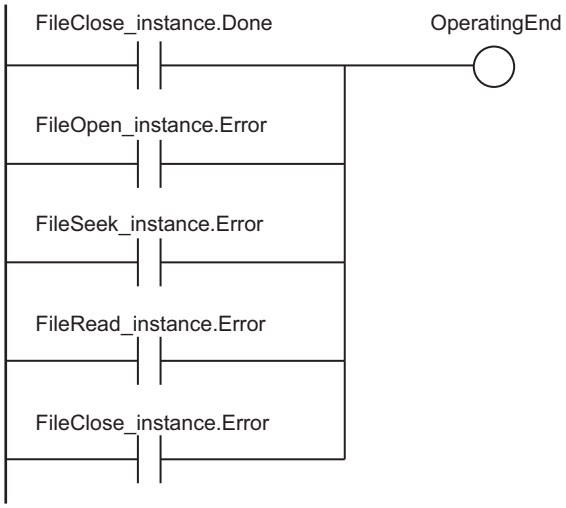
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена.
	Trigger	BOOL	ЛОЖЬ	Условие выполнения

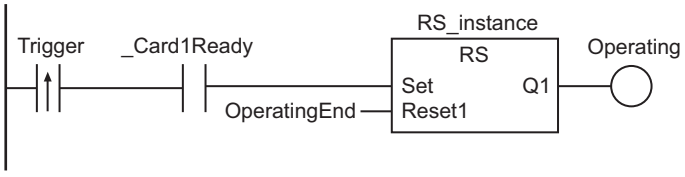
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Operating	BOOL	ЛОЖЬ	Обработка
	Fid	DWORD	16#0	Идентификатор файла
	InDat	ARRAY[0..999] OF BYTE	[1000(16#0)]	Прочитанные данные
	RS_instance	RS		
	FileOpen_instance	FileOpen		
	FileSeek_instance	FileSeek		
	FileRead_instance	FileRead		
	FileClose_instance	FileClose		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

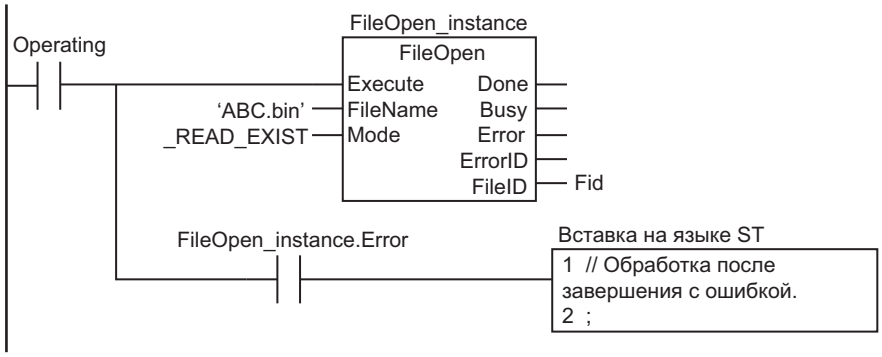
Определение, завершено ли выполнение команды.



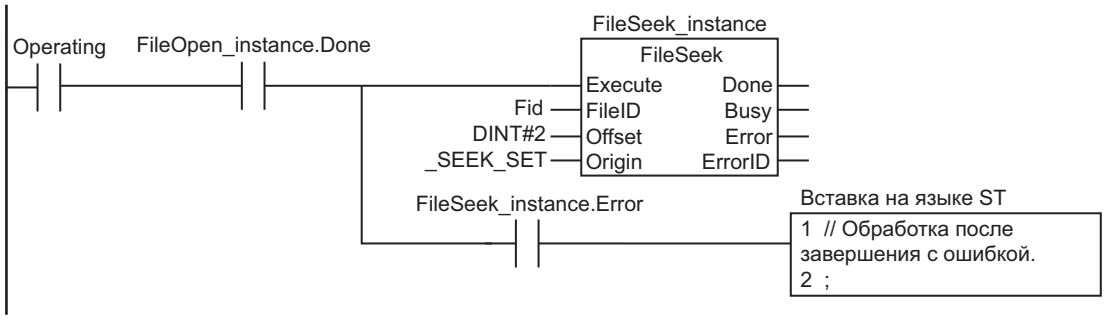
Прием условия выполнения.



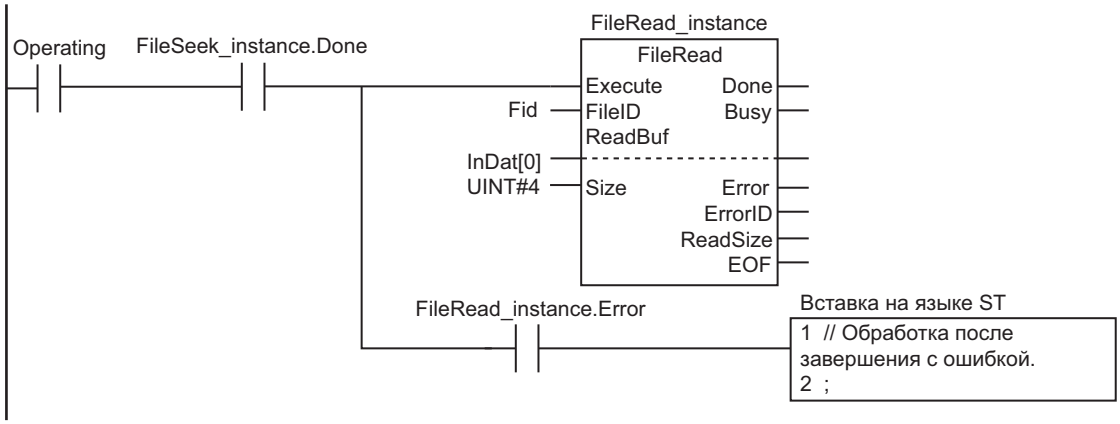
Выполнение команды FileOpen.



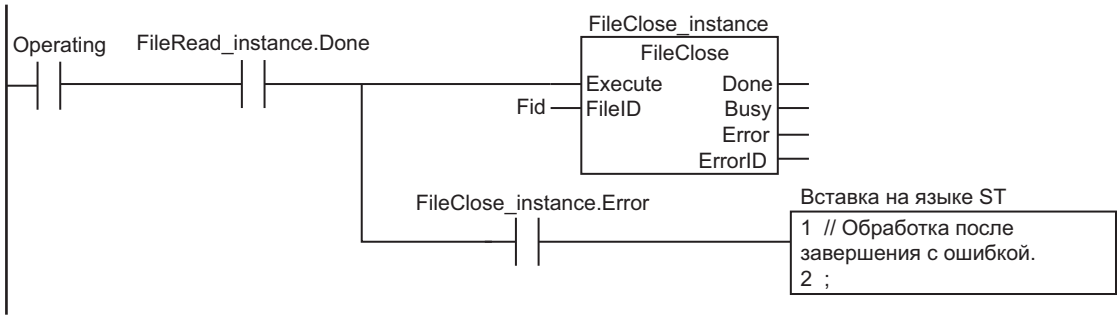
Выполнение команды FileSeek.

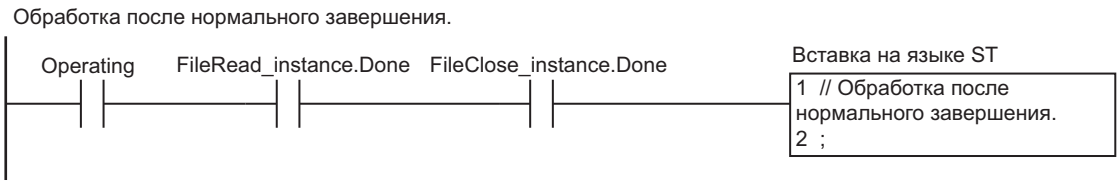


Выполнение команды FileRead.



Выполнение команды FileClose.





Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась.
	Operating	BOOL	ЛОЖЬ	Обработка
	InDat	ARRAY[0..999] OF BYTE	[1000(16#0)]	Прочитанные данные
	Stage	INT	0	Смена этапа
	Fid	DWORD	16#0	Идентификатор файла
	FileOpen_instance	FileOpen		
	FileSeek_instance	FileSeek		
	FileRead_instance	FileRead		
	FileClose_instance	FileClose		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

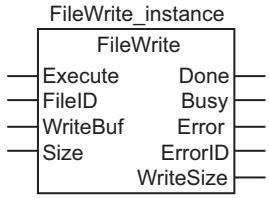
```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация экземпляра.
IF (OperatingStart=TRUE) THEN
    FileOpen_instance(Execute:=FALSE);     // Инициализация экземпляра.
    FileSeek_instance(Execute:=FALSE);     // Инициализация экземпляра.
    FileRead_instance(
        Execute:=FALSE,                    // Инициализация экземпляра.
        ReadBuf:=InDat[0]);                // Фикция
    FileClose_instance(Execute:=FALSE);    // Инициализация экземпляра.
    Stage         :=INT#1;
```

```
    OperatingStart:=FALSE;
END_IF;

// Выполнение команд.
IF (Operating=TRUE) THEN
    CASE Stage OF
    1 :                         // Открытие файла.
        FileOpen_instance(
            Execute :=TRUE,
            FileName:='ABC.bin',        // Имя файла
            Mode    :=_READ_EXIST,        // Прочитать файл.
            FileID  =>Fid);               // Идентификатор файла

        IF (FileOpen_instance.Done=TRUE) THEN
            Stage:=INT#2;                   // Нормальное завершение
        END_IF;

        IF (FileOpen_instance.Error=TRUE) THEN
            Stage:=INT#99;                  // Завершение с ошибкой
        END_IF;
    2 :                         // Поиск в файле.
        FileSeek_instance(
            Execute:=TRUE,
            FileID :=Fid,                   // Идентификатор файла
            Offset :=DINT#2,                // Указатель позиции в файле переходит
на второй байт от начала.
            Origin :=_SEEK_SET);            //

        IF (FileSeek_instance.Done=TRUE) THEN
            Stage:=INT#3;                   // Нормальное завершение
        END_IF;

        IF (FileSeek_instance.Error=TRUE) THEN
            Stage:=INT#99;                  // Завершение с ошибкой
        END_IF;

    3 :                         // Прочитать из файла.
        FileRead_instance(
            Execute:=TRUE,
            FileID :=Fid,                   // Идентификатор файла
            ReadBuf:=InDat[0],              // Буфер чтения
            Size   :=UINT#4);               // Количество элементов для чтения: 4 б
айта

        IF (FileRead_instance.Done=TRUE) THEN
            Stage:=INT#4;             // Нормальное завершение
        END_IF;

        IF (FileRead_instance.Error=TRUE) THEN
            Stage:=INT#99;            // Завершение с ошибкой
        END_IF;
```

```
    4 :                         // Закрытие файла.
        FileClose_instance(
            Execute:=TRUE,
            FileID :=Fid);                  // Идентификатор файла

        IF (FileClose_instance.Done=TRUE) THEN
            Operating:=FALSE;         // Нормальное завершение
        END_IF;

        IF (FileClose_instance.Error=TRUE) THEN
            Stage:=INT#99;            // Завершение с ошибкой
        END_IF;

    99 :
        Operating:=FALSE;             // Обработка после завершения с ошибкой.
    END_CASE;
END_IF;
```

FileWrite

Команда FileWrite записывает данные в указанный файл на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileWrite	Запись в файл	FB		FileWrite_instance(Execute, FileID, WriteBuf, Size, Done, Busy, Error, ErrorID, WriteSize);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileID	Идентификатор файла	Вход	Идентификатор файла для записи	Зависит от типа данных.	---	0
WriteBuf[] (массив)	Буфер записи		Записываемые данные			*1
Size	Количество записываемых элементов		Количество записываемых элементов			1
WriteSize	Количество записанных элементов	Выход	Количество фактически записанных элементов	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

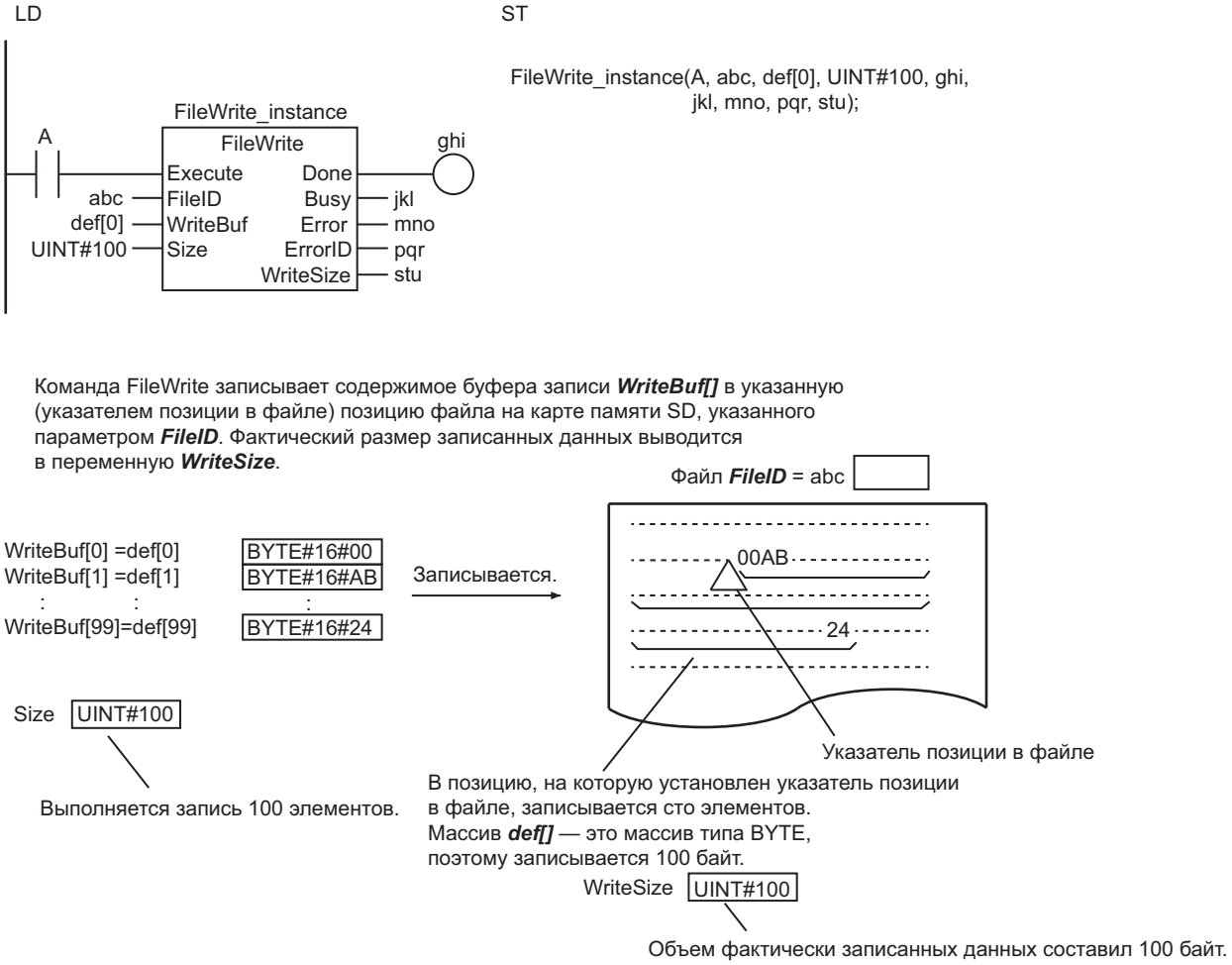
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileID				OK																
WriteBuf[] (массив)	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
	Также допускается указывать массивы перечислений или структур.																			
Size							OK													
WriteSize							OK													

Функция

Команда FileWrite записывает данные в указанную позицию файла на карте памяти SD, указанного параметром *FileID*. Требуемая позиция записи определяется указателем позиции, который устанавливается заранее с помощью команды FileSeek.
В файл записывается содержимое буфера записи WriteBuf[].

Объем записываемых данных в *Size* раз больше, чем размер, соответствующий типу данных переменной WriteBuf[]. Другими словами, он равен объему *Size* элементов массива WriteBuf[]. Для параметра WriteBuf[] можно указать массив перечислений или структур.
Объем фактически записанных данных выводится в переменную *WriteSize*.

Пример программы представлен на рисунке ниже. Если буфер записи def[] содержит данные типа BYTE, в файл записывается 100 байтов данных.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

Имя	Значение	Тип данных	Описание
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.
*2. Подразумевается доступ к карте памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Все данные для записи в команде не сохраняются.
  Согласованность значения может не обеспечиваться, так как значение не синхронизируется с временем выполнения команды и к переменной, передаваемой в WriteBuf[], каждый раз производится обращение.
  Не следует обращаться к целевой переменной во время выполнения команды.
  В случае обращения к переменной во время выполнения команды в файл может быть записано непредусмотренное значение.
- Перед выполнением этой команды необходимо использовать команду FileOpen для получения значения *FileID*.
- Данные записываются побайтово. Младшие байты записываются перед старшими байтами (прямой порядок байтов).
- Если WriteBuf[] является массивом структур, между членами структур могут вставляться дополнительные согласующие области в зависимости от состава структуры.
- Если во время выполнения этой команды модуль ЦПУ перейдет в режим «Программирование» или возникнет ошибка контроллера критического уровня, файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- Даже если для записи данных на карту памяти SD используется команда FileWrite (Запись в файл) и команда завершается нормально, данные могут быть записаны на карту памяти SD неправильно.
  Если нужно быть уверенным, что данные на карту памяти SD были записаны правильно, в пользовательской программе следует предусмотреть, чтобы записанные данные считывались с помощью команды FileRead (Чтение из файла) и сравнивались с исходными данными.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.

a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
b) Карта памяти SD защищена от записи.
c) На карте памяти SD недостаточно свободного места.
d) Количество элементов массива в WriteBuf[] меньше значения *Size*.
e) Файла с указанным идентификатором *FileID* не существует.
f) К файлу с указанным идентификатором *FileID* в данный момент производится доступ.
g) Файл, указанный идентификатором *FileID*, не был открыт в режиме записи.
h) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

В данном случае записываются четыре байта данных в файл «ABC.bin», начиная со второго байта от начала файла. В файл записывается содержимое переменной-массива OutDat[] типа BYTE.
Обработка выполняется в следующем порядке:

1 Используется команда FileOpen для открытия файла «ABC.bin».

2 С помощью команды FileSeek указатель позиции в файле устанавливается на второй байт от начала файла.

3 Используется команда FileWrite для записи четырех байтов данных из переменной-массива OutDat[] в указанную позицию в файле.

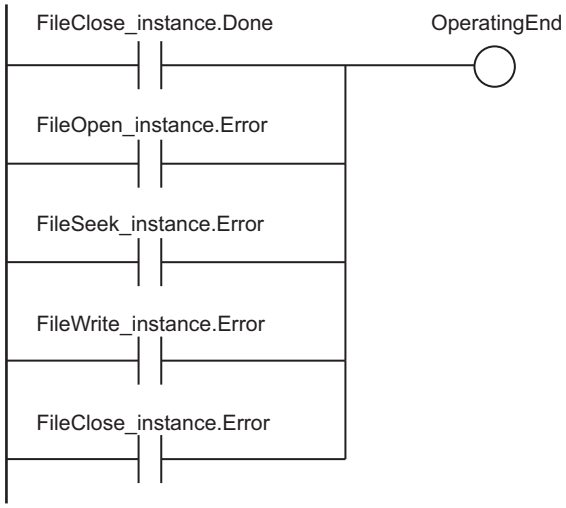
4 Используется команда FileClose для закрытия файла «ABC.bin».

Программа на языке LD

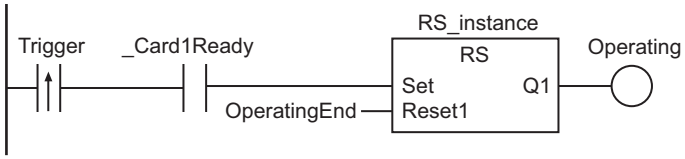
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена.
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	Fid	DWORD	16#0	Идентификатор файла
	OutDat	ARRAY[0..999] OF BYTE	[1000(16#0)]	Записываемые данные
	RS_instance	RS		
	FileOpen_instance	FileOpen		
	FileSeek_instance	FileSeek		
	FileWrite_instance	FileWrite		
	FileClose_instance	FileClose		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

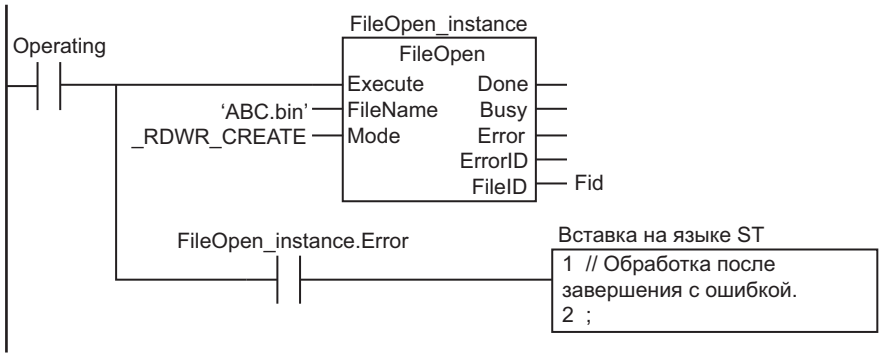
Определение, завершено ли выполнение команды.



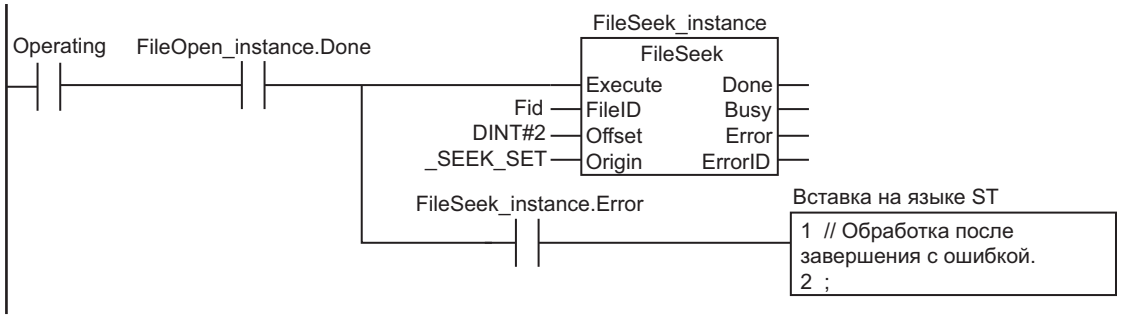
Прием условия выполнения.



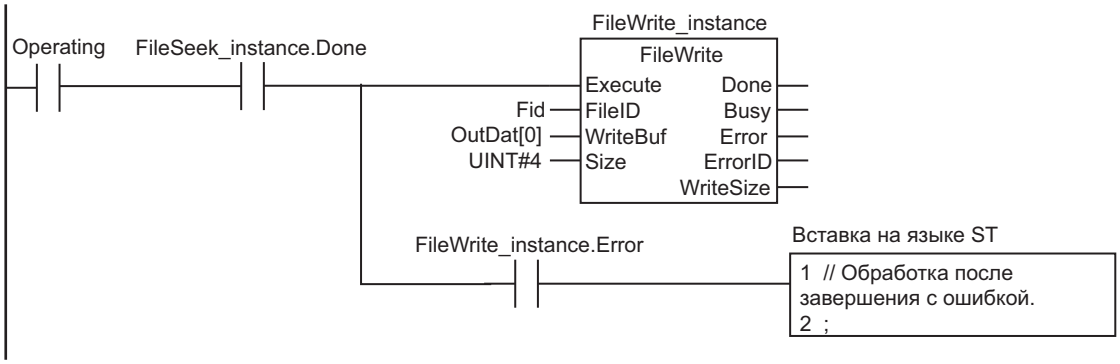
Выполнение команды FileOpen.



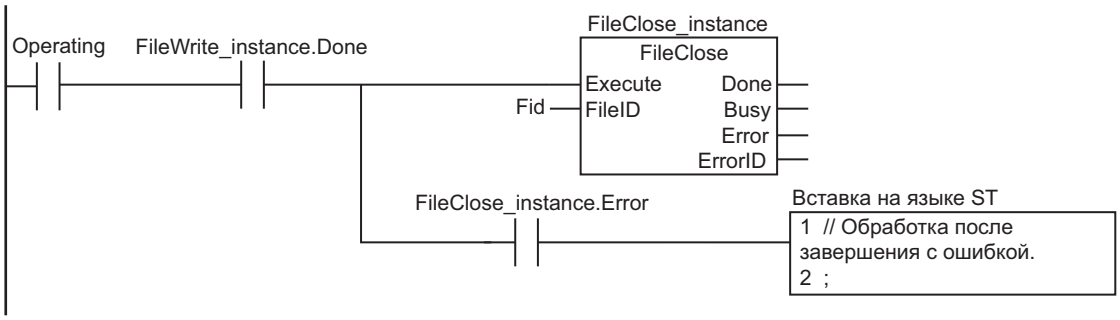
Выполнение команды FileSeek.



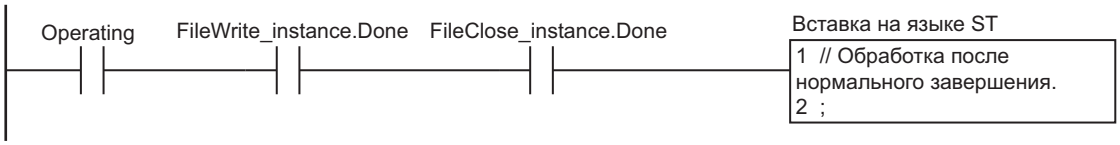
Выполнение команды FileWrite.



Выполнение команды FileClose.



Обработка после нормального завершения.



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась.
	Operating	BOOL	ЛОЖЬ	Обработка
	OutDat	ARRAY[0..999] OF BYTE	[1000(16#0)]	Записываемые данные
	Stage	INT	0	Смена этапа
	Fid	DWORD	16#0	Идентификатор файла
	FileOpen_instance	FileOpen		
	FileSeek_instance	FileSeek		
	FileWrite_instance	FileWrite		

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	FileClose_instance	FileClose		

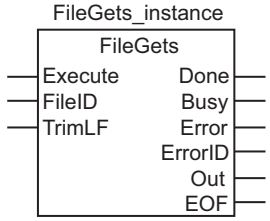
Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация экземпляра.
IF (OperatingStart=TRUE) THEN
    FileOpen_instance(Execute:=FALSE);
    FileSeek_instance(Execute:=FALSE);
    FileWrite_instance(
        Execute :=FALSE,
        WriteBuf:=OutDat[0]);
    FileClose_instance(Execute:=FALSE);
    Stage         :=INT#1;
    OperatingStart:=FALSE;
END_IF;

// Выполнение команд.
IF (Operating=TRUE) THEN
    CASE Stage OF
    1 :                    // Открытие файла.
        FileOpen_instance(
            Execute :=TRUE,
            FileName:='ABC.bin',       // Имя файла
            Mode    :=_RDWR_CREATE,    // Чтение файла и запись.
            FileID  =>Fid);            // Идентификатор файла

        IF (FileOpen_instance.Done=TRUE) THEN
            Stage:=INT#2;         // Нормальное завершение
        END_IF;

        IF (FileOpen_instance.Error=TRUE) THEN
            Stage:=INT#99;        // Завершение с ошибкой
        END_IF;
    2 :                    // Поиск в файле.
        FileSeek_instance(
            Execute:=TRUE,
            FileID :=Fid,              // Идентификатор файла
            Offset :=DINT#2,           // Указатель позиции в файле устанавливается
```

```
 на второй байт от начала.
            Origin :=_SEEK_SET);       //

        IF (FileSeek_instance.Done=TRUE) THEN
            Stage:=INT#3;         // Нормальное завершение
        END_IF;

        IF (FileSeek_instance.Error=TRUE) THEN
            Stage:=INT#99;        // Завершение с ошибкой
        END_IF;

    3 :                    // Запись в файл.
        FileWrite_instance(
            Execute :=TRUE,
            FileID  :=Fid,             // Идентификатор файла
            WriteBuf:=OutDat[0],       // Буфер записи
            Size    :=UINT#4);         // Количество элементов для записи: 4 байта

        IF (FileWrite_instance.Done=TRUE) THEN
            Stage:=INT#4;         // Нормальное завершение
        END_IF;

        IF (FileWrite_instance.Error=TRUE) THEN
            Stage:=INT#99;        // Завершение с ошибкой
        END_IF;

    4 :                    // Закрытие файла.
        FileClose_instance(
            Execute:=TRUE,
            FileID :=Fid);             // Идентификатор файла

        IF (FileClose_instance.Done=TRUE) THEN
            Operating:=FALSE;     // Нормальное завершение
        END_IF;

        IF (FileClose_instance.Error=TRUE) THEN
            Stage:=INT#99;        // Завершение с ошибкой
        END_IF;

    99 :
        Operating:=FALSE;        // Обработка после завершения с ошибкой.
    END_CASE;
END_IF;
```

FileGets

Команда FileGets считывает текстовую строку из одной строки указанного файла на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileGets	Получить текстовую строку	FB		FileGets_instance(Execute, FileID, TrimLF, Done, Busy, Error, ErrorID, Out, EOF);

Переменные

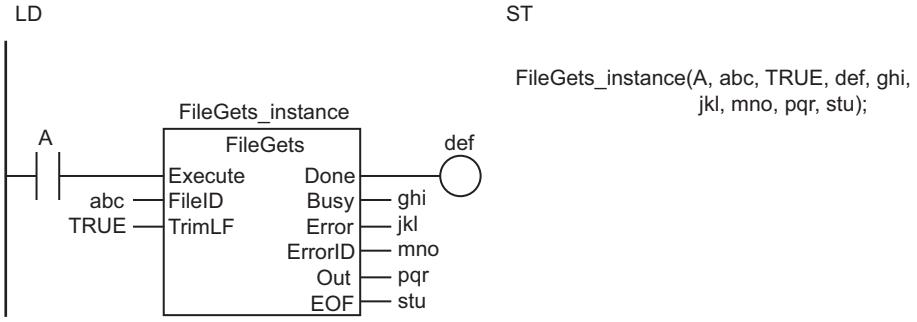
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileID	Идентификатор файла	Вход	Идентификатор читаемого файла	Зависит от типа данных.	---	0
TrimLF	Обработка перевода строки		Обработка кода перевода строки в прочитанной текстовой строке ИСТИНА: удалить. ЛОЖЬ: не удалять.			ЛОЖЬ
Out	Прочитанная текстовая строка	Выход	Текстовая строка, которая была прочитана	Зависит от типа данных.	---	---
EOF	Конец файла		Показывает, достигнут ли конец файла. ИСТИНА: достигнут. ЛОЖЬ: не достигнут.			

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileID				OK																
TrimLF	OK																			
Out																				OK
EOF	OK																			

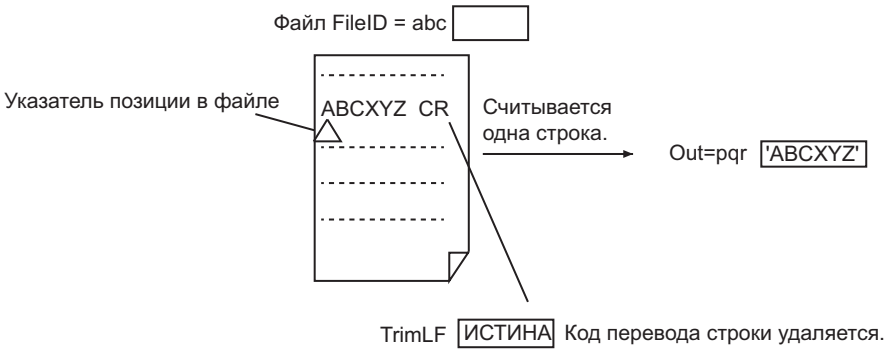
Функция

Команда FileGets считывает текстовую строку из одной строки файла на карте памяти SD, указанного параметром *FileID*, в указанной позиции. Требуемая позиция чтения определяется указателем позиции,
который устанавливается заранее с помощью команды FileSeek.
Признаком конца строки файла служит код перевода строки.
Прочитанная текстовая строка записывается в переменную *Out* (прочитанная текстовая строка).
Автоматически распознаются три следующих кода перевода строки: CR, LF и CR+LF.
Если для параметра *TrimLF* (обработка перевода строки) указано значение ИСТИНА, код перевода строки удаляется из текстовой строки перед записью в *Out*.
Если из файла прочитаны данные до конца файла, значение переменной *EOF* меняется на ИСТИНА. В противном случае в *EOF* остается значение ЛОЖЬ.

Пример программы представлен на рисунке ниже. В данном случае считывается текстовая строка из одной строки файла, из нее удаляется код перевода строки и результат записывается в переменную *pqr*.



Команда FileGets считывает текстовую строку из одной строки файла на карте памяти SD, указанного параметром ***FileID***, в позиции, указанной указателем позиции в файле, и сохраняет ее в переменную ***Out*** (прочитанная текстовая строка). Код перевода строки при этом удаляется.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.

Имя	Значение	Тип данных	Описание
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

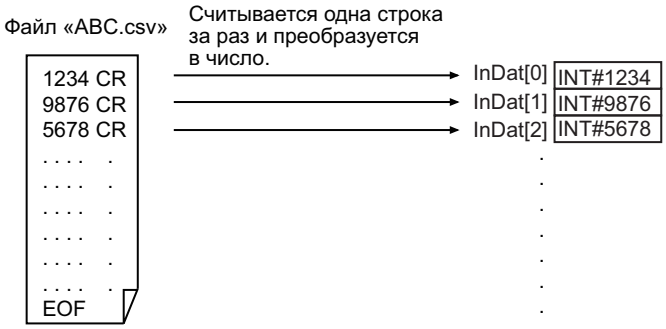
*2. Подразумевается доступ к карте памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если длина текстовой строки в одной строке файла превышает 1986 байтов (для кодировки UTF-8, с учетом конечного пустого символа (NULL)), в *Out* сохраняются первые 1985 байтов текстовой строки и к ним добавляется пустой символ (NULL).
- Перед выполнением этой команды необходимо использовать команду FileOpen для получения значения *FileID*.
- Если во время выполнения этой команды модуль ЦПУ перейдет в режим «Программирование» или возникнет ошибка контроллера критического уровня, файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Файла с указанным идентификатором *FileID* не существует.
  c) К файлу с указанным идентификатором *FileID* в данный момент производится доступ.
  d) Файл, указанный идентификатором *FileID*, не был открыт в режиме чтения.
  e) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

В данном примере обрабатывается файл с именем «ABC.csv», который содержит несколько текстовых строк, разделяемых кодами CR. Все эти текстовые строки состоят из чисел. Строки считываются из файла по одной, каждая текстовая строка преобразуется в соответствующее целое число, и результаты сохраняются в переменную-массив InDat[] типа INT. Обработка завершается, когда оказываются прочитаны все данные до конца файла. Предполагается, что рассматриваемая программа выполняется в рамках циклической задачи.



Обработка выполняется в следующем порядке:

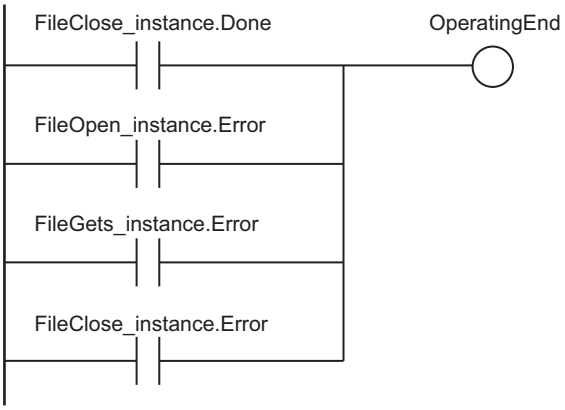
1. Используется команда FileOpen для открытия файла «ABC.csv».
2. Используется команда FileGets для чтения одной строки из файла.
3. С помощью команды STRING_TO_INT прочитанная текстовая строка преобразуется в целое число и сохраняется в переменную InDat[].
4. Шаги 2 и 3 повторяются до достижения кода EOF (конца файла).
5. Используется команда FileClose для закрытия файла.

Программа на языке LD

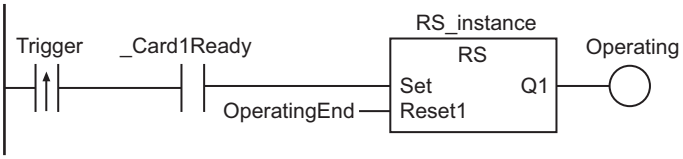
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена.
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	Index	INT	0	Индекс элемента InDat[]
	Fid	DWORD	16#0	Идентификатор файла
	InDat	ARRAY[0..999] OF INT	[1000(0)]	Целочисленные данные
	RS_instance	RS		
	FileOpen_instance	FileOpen		
	FileGets_instance	FileGets		
	FileClose_instance	FileClose		

Внешние перемен-ные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

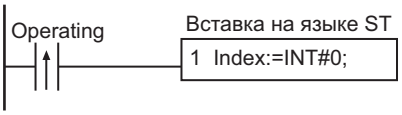
Определение, завершено ли выполнение команды.



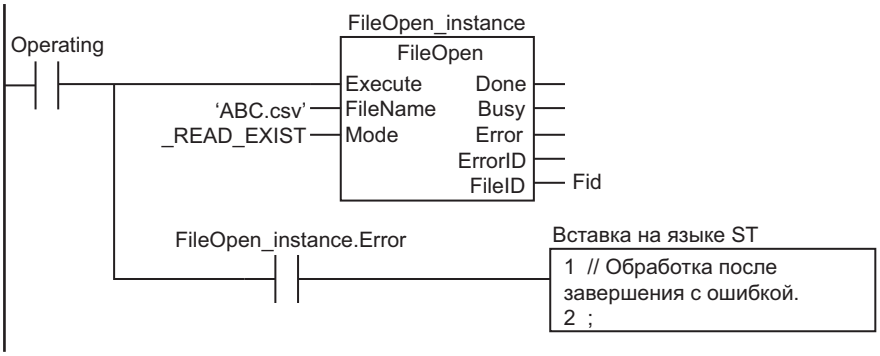
Прием условия выполнения.



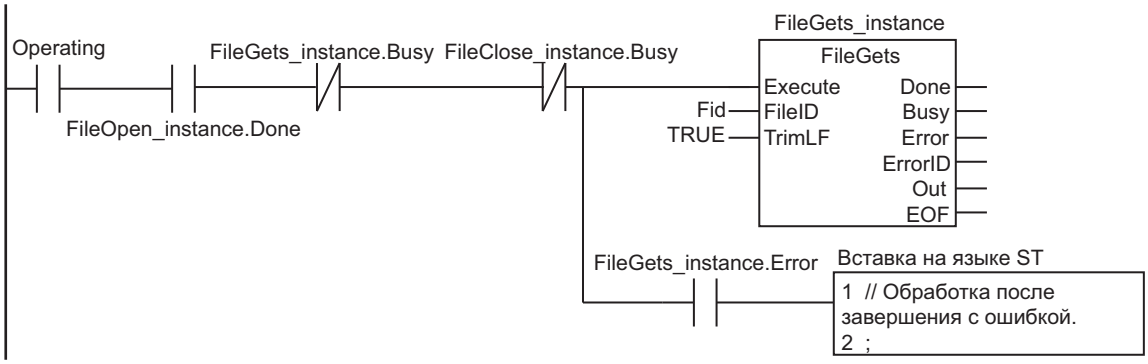
Инициализация индекса элемента ***InDat[]***.



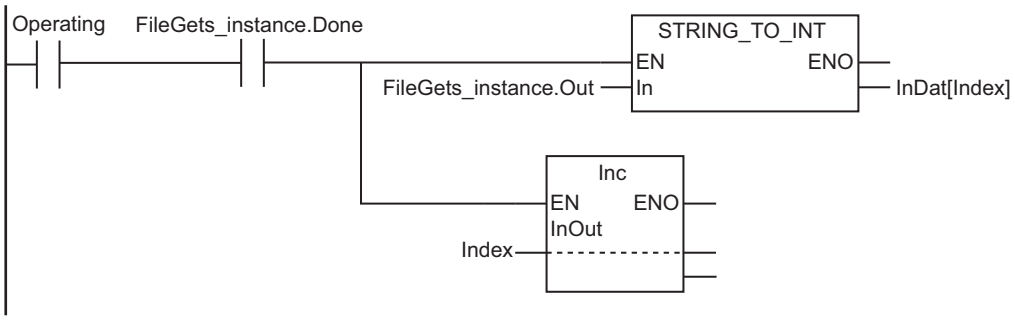
Выполнение команды FileOpen.



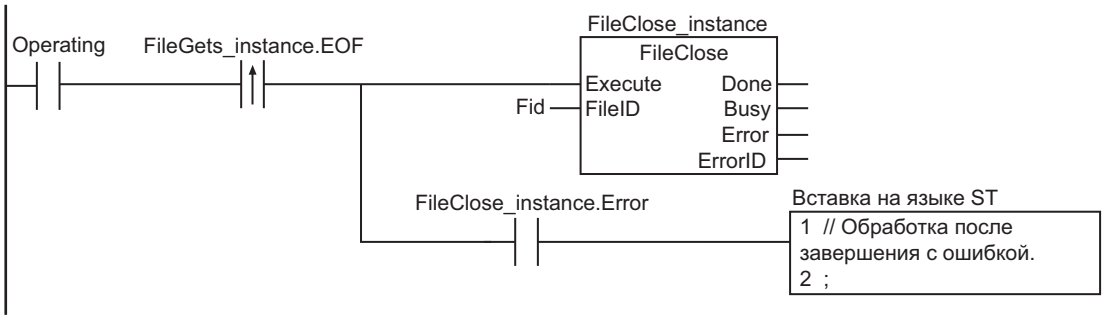
Выполнение команды FileGets.



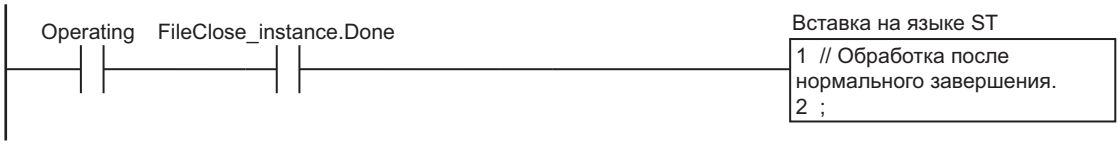
Выполнение команды STRING_TO_INT.



Выполнение команды FileClose при обнаружении EOF.



Обработка после нормального завершения.



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась.
	Operating	BOOL	ЛОЖЬ	Обработка
	InDat	ARRAY[0..999] OF INT	[1000(0)]	Целочисленные данные
	Stage	INT	0	Смена этапа
	Index	INT	0	Индекс элемента InDat[]
	Fid	DWORD	16#0	Идентификатор файла
	FileOpen_instance	FileOpen		
	FileGets_instance	FileGets		
	FileClose_instance	FileClose		

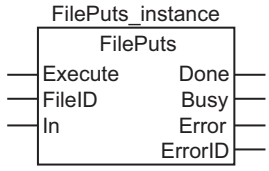
Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация экземпляра.
IF (OperatingStart=TRUE) THEN
    FileOpen_instance(Execute:=FALSE);
    FileGets_instance(Execute:=FALSE);
    FileClose_instance(Execute:=FALSE);
    Stage         :=INT#1;
    Index         :=INT#0;
    OperatingStart:=FALSE;
END_IF;

// Выполнение команды.
IF (Operating=TRUE) THEN
    CASE Stage OF
    1 :                  // Открытие файла.
        FileOpen_instance(
            Execute :=TRUE,
            FileName:='ABC.csv',     // Имя файла
            Mode    :=_READ_EXIST,   // Чтение файла.
            FileID  =>Fid);          // Идентификатор файла

        IF (FileOpen_instance.Done=TRUE) THEN
            Stage:=INT#2;       // Нормальное завершение
        END_IF;
```

```
        IF (FileOpen_instance.Error=TRUE) THEN
            Stage:=INT#99;      // Завершение с ошибкой
        END_IF;
    2 :                 // Чтение текстовой строки.
        FileGets_instance(
            Execute:=TRUE,
            FileID :=Fid,
            TrimLF :=TRUE);

        IF (FileGets_instance.Done=TRUE) THEN
            // Преобразование прочитанной текстовой строки в целое число.
            InDat[Index]:=STRING_TO_INT(FileGets_instance.Out);
            Index:=Index+INT#1;

            // Достигнут конец файла.
            IF (FileGets_instance.EOF=TRUE) THEN
                Stage:=INT#3;   // Нормальное завершение
            ELSE
                FileGets_instance(Execute:=FALSE);
            END_IF;
        END_IF;

        IF (FileGets_instance.Error=TRUE) THEN
            Stage:=INT#99;      // Завершение с ошибкой
        END_IF;

    3 :                 // Закрытие файла.
        FileClose_instance(
            Execute:=TRUE,
            FileID :=Fid);          // Идентификатор файла

        IF (FileClose_instance.Done=TRUE) THEN
            Operating:=FALSE;   // Нормальное завершение
        END_IF;

        IF (FileClose_instance.Error=TRUE) THEN
            Stage:=INT#99;      // Завершение с ошибкой
        END_IF;

    99 :                // Обработка после завершения с ошибкой.
        Operating:=FALSE;
    END_CASE;
END_IF;
```

FilePuts

Команда FilePuts записывает текстовую строку в указанный файл на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FilePuts	Поместить текстовую строку	FB		FilePuts_instance(Execute, FileID, In, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileID	Идентификатор файла	Вход	Идентификатор файла для записи	Зависит от типа данных.	---	0
In	Записываемая текстовая строка		Текстовая строка для записи			"

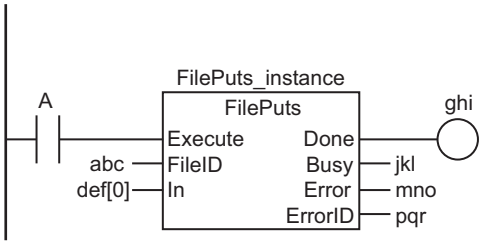
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileID				OK																
In																				OK

Функция

Команда FilePuts записывает текстовую строку в указанную позицию файла на карте памяти SD, указанного параметром *FileID*. Требуемая позиция записи определяется указателем позиции, который устанавливается заранее с помощью команды FileSeek.
В файл записывается содержимое текстовой строки для записи *In*.

Пример программы представлен на рисунке ниже. В данном случае в файл записывается содержимое элемента массива def[0].

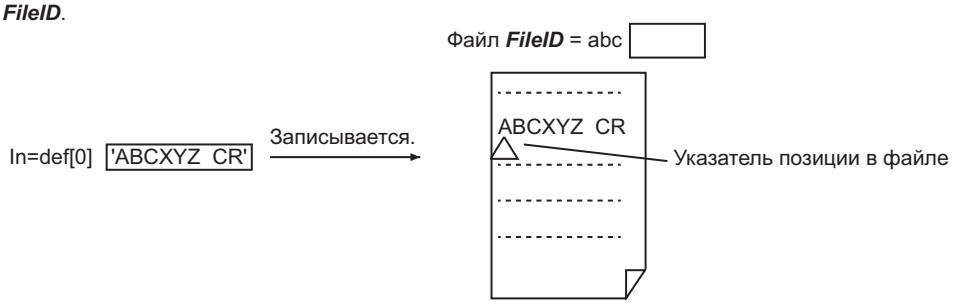
LD



ST

```
FilePuts_instance(A, abc, def[0], ghi, jkl, mno, pqr);
```

Команда FilePuts записывает содержимое ***In*** (текстовая строка для записи) в указанную (указателем позиции в файле) позицию файла на карте памяти SD, указанного параметром ***FileID***.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

Если после записи текстовой строки должен производиться перевод строки, в конец содержимого *In* следует добавить код перевода строки.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Перед выполнением этой команды необходимо использовать команду FileOpen для получения значения *FileID*.
- Если во время выполнения этой команды модуль ЦПУ перейдет в режим «Программирование» или возникнет ошибка контроллера критического уровня, файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- Даже если для записи данных на карту памяти SD используется команда FilePuts (Поместить текстовую строку) и команда завершается нормально, данные могут быть записаны на карту памяти SD неправильно.
  Если нужно быть уверенным, что данные на карту памяти SD были записаны правильно, в пользовательской программе следует предусмотреть, чтобы записанные данные считывались с помощью команды FileGets (Получить текстовую строку) и сравнивались с исходными данными.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Карта памяти SD защищена от записи.
  c) На карте памяти SD недостаточно свободного места.
  d) Файла с указанным идентификатором *FileID* не существует.
  e) К файлу с указанным идентификатором *FileID* в данный момент производится доступ.
  f) Файл, указанный идентификатором *FileID*, не был открыт в режиме записи.
  g) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

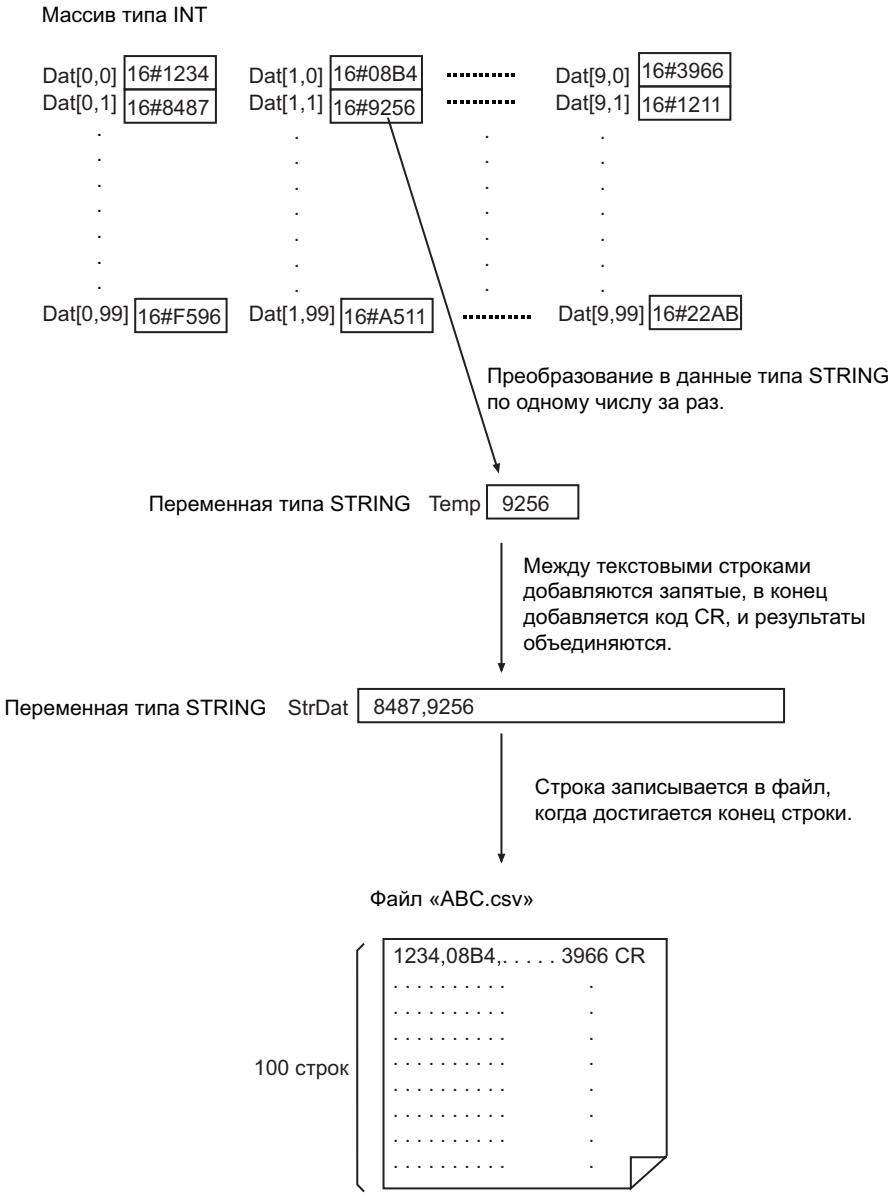
В данном примере 100 строк содержимого переменной-массива Dat[0..9,0..99] типа INT сохраняются в файл с именем «ABC.csv» в формате CSV.
Каждая строка файла содержит 10 текстовых строк с числами. Между ними вставляются запятые. В конец каждой строки файла добавляется код CR+LF.
Соблюдается следующий порядок действий:

1 Элемент массива Dat[] преобразуется в текстовую строку и сохраняется в переменную *Temp* типа STRING.

2 Если конец формируемой строки файла еще не достигнут, добавляется запятая и содержимое *Temp* присоединяется к переменной *StrDat* типа STRING. Если это последнее значение в формируемой строке файла, добавляется код CR+LF для завершения строки и содержимое *Temp* добавляется к переменной *StrDat* типа STRING.

3 Когда строка завершена, содержимое *StrDat* записывается в файл.

4 Шаги 1–3 повторяются для 100 строк файла.



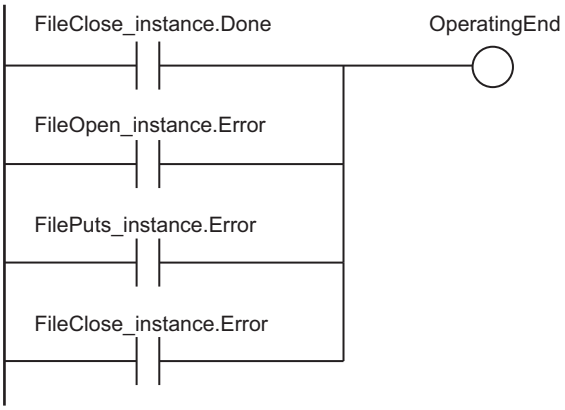
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена.

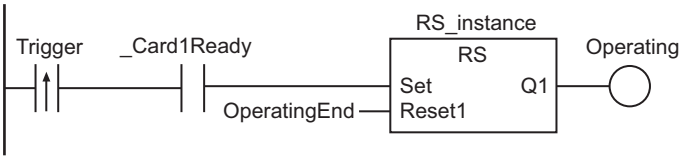
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	Index0	INT	0	Индекс столбца
	Index1	INT	0	Индекс строки
	Fid	DWORD	16#0	Идентификатор файла
	StrDat	STRING[255]	''	Данные текстовой строки
	Dat	ARRAY[0..99,0..9] OF INT	[1000(0)]	Числовые значения
	Temp	STRING[255]	''	Временные данные
	RS_instance	RS		
	FileOpen_instance	FileOpen		
	FilePuts_instance	FilePuts		
	FileClose_instance	FileClose		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

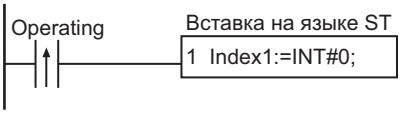
Определение, завершено ли выполнение команды.



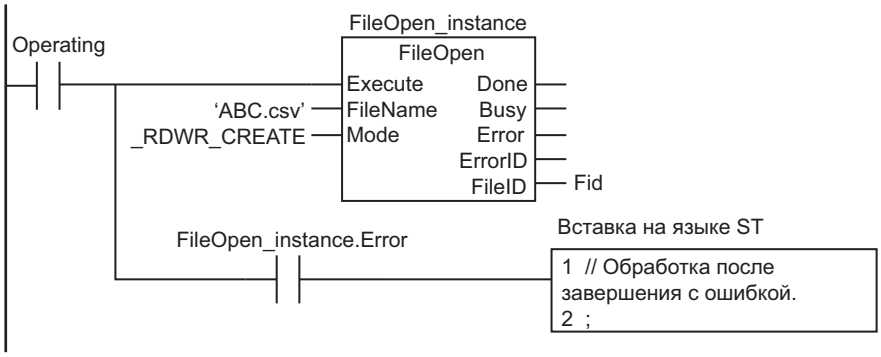
Прием условия выполнения.



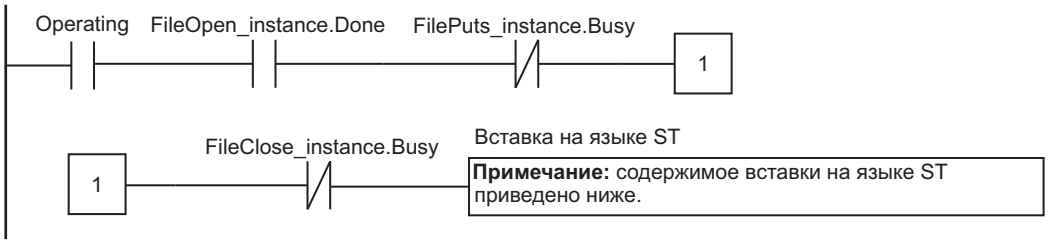
Инициализация индекса строки.



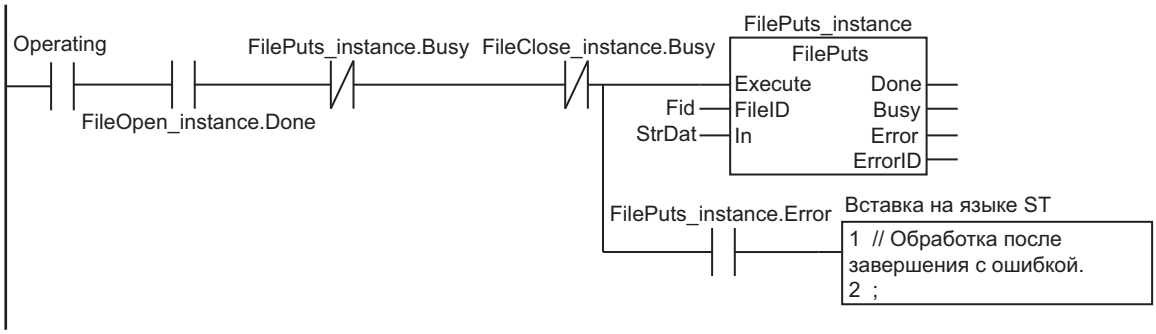
Выполнение команды FileOpen.



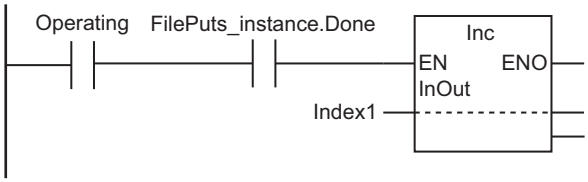
Создание текстовой строки для одной строки файла.



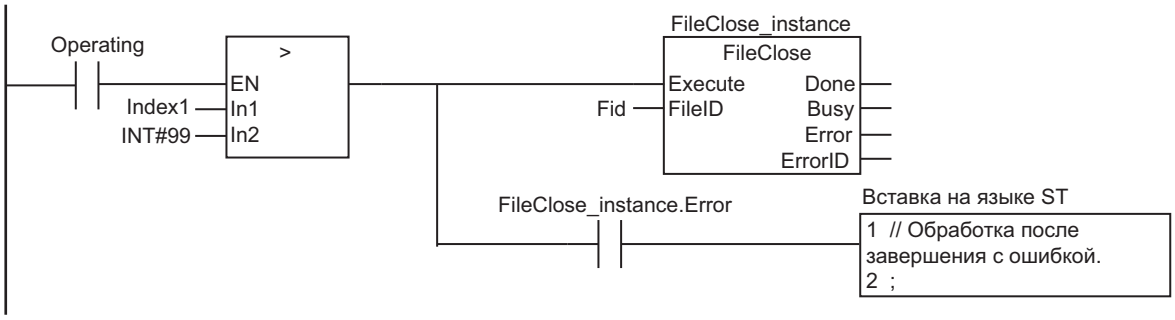
Запись текстовой строки для одной строки в файл.

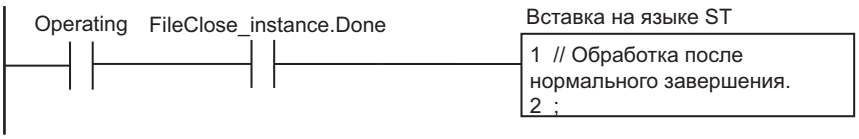


Увеличение индекса строки на 1.



Выполнить команду FileClose после того, как будет записано 100 строк.





● Содержимое вставки на языке ST

```
StrDat:='';

// Конкатенация текстовых строк 0...8.

FOR Index0:=INT#0 TO INT#8 BY INT#1 DO
    Temp  :=INT_TO_STRING(Dat[Index1, Index0]);
    Temp  :=CONCAT(In1:=Temp, In2:=',');
    StrDat:=CONCAT(In1:=StrDat, In2:=Temp);
END_FOR;

// Присоединение текстовой строки 9 и добавление CR+LF.
Temp  :=INT_TO_STRING(Dat[Index1, Index0]);
Temp  :=CONCAT(In1:=Temp, In2:='$r$l');
StrDat:=CONCAT(In1:=StrDat, In2:=Temp);
```

Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение Trigger в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась.
	Operating	BOOL	ЛОЖЬ	Обработка
	Stage	INT	0	Смена этапа
	Index0	INT	0	Индекс столбца
	Index1	INT	0	Индекс строки
	Fid	DWORD	16#0	Идентификатор файла
	StrDat	STRING[255]	''	Данные текстовой строки
	Dat	ARRAY[0..99,0..9] OF INT	[1000(0)]	Числовые значения
	Temp	STRING[255]	''	Временные данные
	FileOpen_instance	FileOpen		
	FilePuts_instance	FilePuts		
	FileClose_instance	FileClose		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация экземпляра.
IF (OperatingStart=TRUE) THEN
    FileOpen_instance(Execute:=FALSE);
    FilePuts_instance(Execute:=FALSE);
    FileClose_instance(Execute:=FALSE);
    Stage         :=INT#1;
    Index1        :=INT#0;       // Инициализация индекса строки.
    OperatingStart:=FALSE;
END_IF;

// Выполнение команд.
IF (Operating=TRUE) THEN
    CASE Stage OF
    1 :                   // Открытие файла.
        FileOpen_instance(
            Execute :=TRUE,
            FileName:='ABC.csv',      // Имя файла
            Mode    :=_RDWR_CREATE,   // Чтение файла
            FileID  =>Fid);           // Идентификатор файла

        IF (FileOpen_instance.Done=TRUE) THEN
            Stage:=INT#2;        // Нормальное завершение
        END_IF;

        IF (FileOpen_instance.Error=TRUE) THEN
            Stage:=INT#99;       // Завершение с ошибкой
        END_IF;
    2 :                   // Создание текстовой строки для одной строки файла.
        StrDat:='';

        // Конкатенация текстовых строк 0...8.
        FOR Index0:=INT#0 TO INT#8 BY INT#1 DO
            Temp   :=INT_TO_STRING(Dat[Index1, Index0]);
            Temp   :=CONCAT(In1:=Temp, In2:=',');
            StrDat :=CONCAT(In1:=StrDat, In2:=Temp);
        END_FOR;

        // Присоединение текстовой строки 9 и добавление CR+LF.
        Temp  :=INT_TO_STRING(Dat[Index1, Index0]);
```

```
        Temp  :=CONCAT(In1:=Temp, In2:='$r$l');
        StrDat:=CONCAT(In1:=StrDat, In2:=Temp);

        Stage:=INT#3;

    3 :                   // Запись текстовой строки.
        FilePuts_instance(
            Execute:=TRUE,
            FileID :=Fid,
            In     :=StrDat);

        IF (FilePuts_instance.Done=TRUE) THEN
            Index1:=Index1+INT#1;

            IF (Index1>INT#99) THEN   // Если было записано 100 строк.
                Stage:=INT#4;
            ELSE
                FilePuts_instance(Execute:=FALSE);
                Stage:=INT#2;
            END_IF;
        END_IF;

        IF (FilePuts_instance.Error=TRUE) THEN
            Stage:=INT#99;       // Завершение с ошибкой
        END_IF;

    4 :                   // Закрытие файла.
        FileClose_instance(
            Execute:=TRUE,
            FileID :=Fid);            // Идентификатор файла

        IF (FileClose_instance.Done=TRUE) THEN
            Operating:=FALSE;    // Нормальное завершение
        END_IF;

        IF (FileClose_instance.Error=TRUE) THEN
            Stage:=INT#99;       // Завершение с ошибкой
        END_IF;

    99 :                  // Обработка после завершения с ошибкой.
        Operating:=FALSE;
    END_CASE;
END_IF;
```

FileCopy

Команда FileCopy создает копию указанного файла на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileCopy	Копирование файла	FB	FileCopy_instance FileCopy Execute Done SrcFileName Busy DstFileName Error OverWrite ErrorID	FileCopy_instance(Execute, SrcFileName, DstFileName, OverWrite, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
SrcFileName	Исходный файл	Вход	Имя копируемого файла	Макс. 66 байт (65 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	"
DstFileName	Целевой файл		Имя целевого файла			
OverWrite	Разрешение перезаписи		ИСТИНА: разрешить перезапись. ЛОЖЬ: запретить перезапись.	Зависит от типа данных.		ЛОЖЬ

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SrcFileName																				ОК
DstFileName																				ОК
OverWrite	ОК																			

Функция

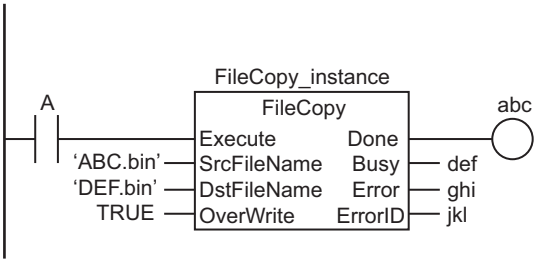
Команда FileCopy создает копию файла, имя которого указано параметром *SrcFileName* (исходный файл), на карте памяти SD. Имя создаваемой копии указывается параметром *DstFileName* (целевой файл).

Если файл с именем *DstFileName* уже существует на карте памяти SD, выполняется одна из следующих операций в зависимости от значения параметра *OverWrite* (разрешение перезаписи).

Значение *OverWrite*	Описание
ИСТИНА (разрешить перезапись.)	Существующий файл будет перезаписан.
ЛОЖЬ (запретить перезапись.)	Файл не перезаписывается, и возникает ошибка.

Пример программы представлен на рисунке ниже. В данном случае файл «DEF.bin» перезаписывается файлом «ABC.bin».

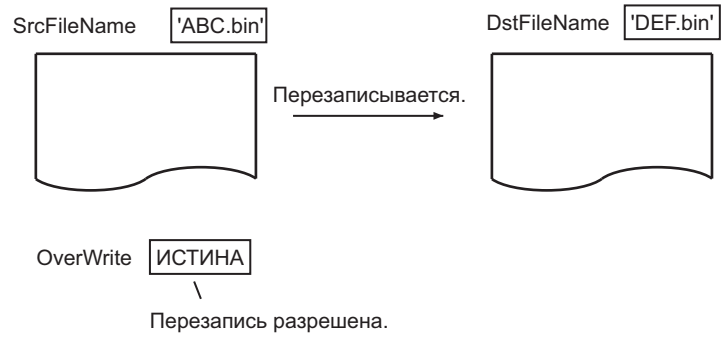
LD



ST

```
FileCopy_instance(A, 'ABC.bin', 'DEF.bin',
                  TRUE, abc, def, ghi, jkl);
```

Команда FileCopy перезаписывает файл, указанный параметром ***SrcFileName*** (исходный файл), файлом, который указан параметром ***DstFileName*** (целевой файл), на карте памяти SD.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

Имя	Значение	Тип данных	Описание
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

Корневым каталогом в полном имени файла является каталог верхнего уровня карты памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если операция копирования завершится ошибкой, на карте памяти SD может остаться неполный файл с именем, которое указано в *DstFileName*.
- При переходе модуля ЦПУ в режим «Программирование» и при возникновении ошибки контроллера критического уровня любой открытый файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Если в то время, когда файл открыт, будет отключено питание нажатием кнопки питания карты памяти SD, файл поврежден не будет.
- Если в то время, когда файл открыт, карта памяти SD будет извлечена из гнезда без нажатия кнопки питания карты памяти SD, содержимое этого файла может быть повреждено. Обязательно выключайте питание карты памяти SD перед ее извлечением.
- Если в то время, когда файл открыт, будет отключено питание или карта памяти SD будет извлечена из гнезда, операции чтения и записи для этого файла будут невозможны, даже если карта памяти SD вновь будет вставлена.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Карта памяти SD защищена от записи.
  c) На карте памяти SD недостаточно свободного места.
  d) Файла с указанным именем *SrcFileName* не существует.

e) Значение *SrcFileName* не является допустимым именем файла.
f) К файлу с указанным именем *SrcFileName* или *DstFileName* в данный момент производится доступ.
g) Значение *DstFileName* не является допустимым именем файла.
h) Файл с именем *DstFileName* уже существует, и значение *OverWrite* равно ЛОЖЬ.
i) Файл с именем *DstFileName* уже существует и защищен от записи.
j) Длина значения *DstFileName* превышает максимально допустимое количество байтов в имени файла.
k) Превышено максимально допустимое количество файлов или каталогов.
l) Выполняются одновременно пять или больше перечисленных ниже команд для работы с картой памяти SD, не имеющих параметра *FileID*: FileWriteVar, FileReadVar, FileCopy, DirCreate, FileRemove, DirRemove и FileRename.
m) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

Рассмотрим реализацию описанной ниже процедуры для перемещения файла.

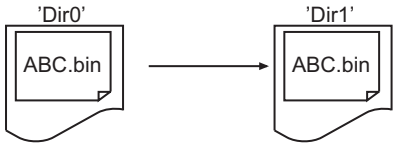
1 С помощью команды DirCreate на карте памяти SD создается каталог с именем «Dir1».

2 Используется команда FileCopy для копирования файла с именем «ABC.bin» в существующем каталоге «Dir0» в каталог «Dir1».

3 Используется команда DirRemove для удаления каталога «Dir0» (источник копии).

1. Создание каталога.

2. Копирование файла.



3. Удаление каталога.

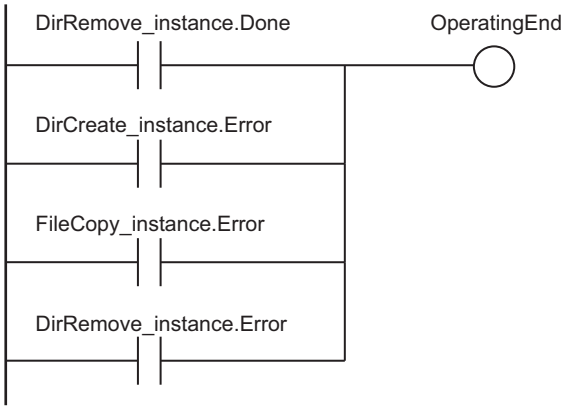
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена.
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка

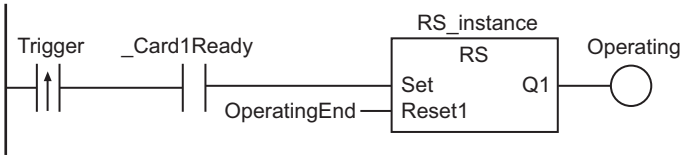
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	RS_instance	RS		
	DirCreate_instance	DirCreate		
	FileCopy_instance	FileCopy		
	DirRemove_instance	DirRemove		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

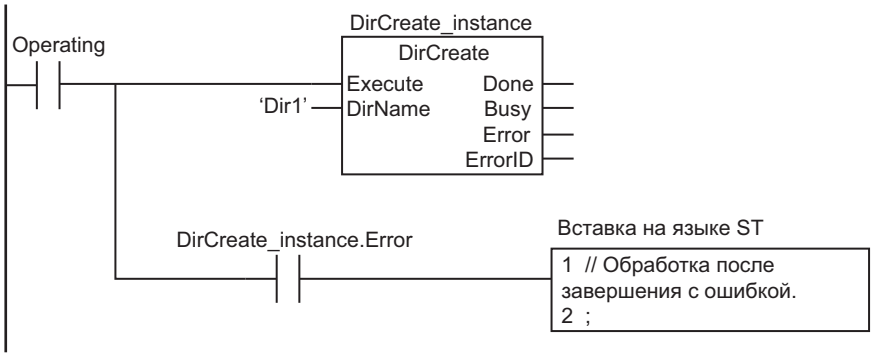
Определение, завершено ли выполнение команды.



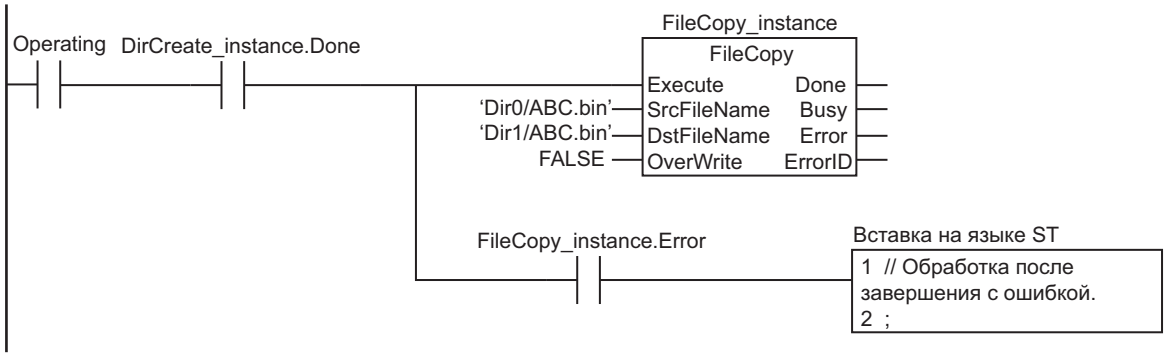
Прием условия выполнения.



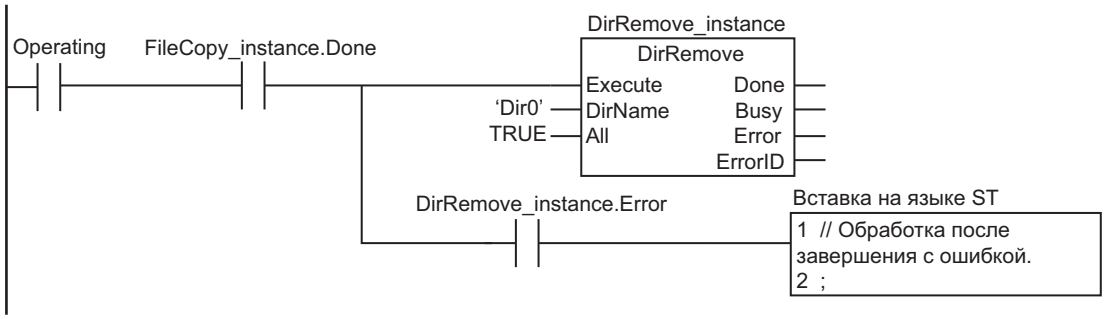
Выполнение команды DirCreate.



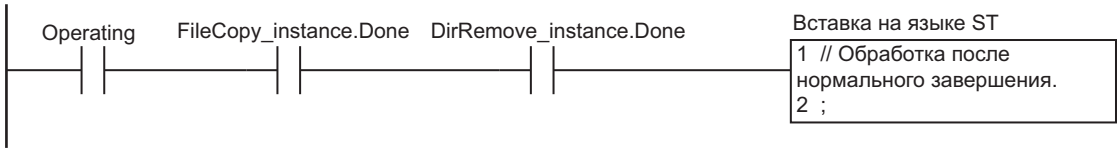
Выполнение команды FileCopy.



Выполнение команды DirRemove.



Обработка после нормального завершения.



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение *Trigger* в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась.
	Operating	BOOL	ЛОЖЬ	Обработка
	Stage	INT	0	Смена этапа
	DirCreate_instance	DirCreate		
	FileCopy_instance	FileCopy		
	DirRemove_instance	DirRemove		

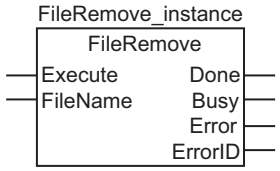
Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация экземпляра.
IF (OperatingStart=TRUE) THEN
    DirCreate_instance(Execute:=FALSE);
    FileCopy_instance(Execute:=FALSE);
    DirRemove_instance(Execute:=FALSE);
    Stage         :=INT#1;
    OperatingStart:=FALSE;
END_IF;

// Выполнение команд.
IF (Operating=TRUE) THEN
    CASE Stage OF
    1 :                             // Создание каталога.
        DirCreate_instance(
            Execute:=TRUE,
            DirName:='Dir1');              // Имя каталога

        IF (DirCreate_instance.Done=TRUE) THEN
            Stage:=INT#2;                  // Нормальное завершение
        END_IF;

        IF (DirCreate_instance.Error=TRUE) THEN
            Stage:=INT#99;                 // Завершение с ошибкой
        END_IF;
    2 :                             // Копирование файла.
        FileCopy_instance(
            Execute    :=TRUE,
            SrcFileName:='Dir0/ABC.bin',   // Имя файла для копирования
            DstFileName:='Dir1/ABC.bin',   // Имя конечного файла
            OverWrite  :=FALSE);           // Запретить перезапись.

        IF (FileCopy_instance.Done=TRUE) THEN
            Stage:=INT#3;
        END_IF;

        IF (FileCopy_instance.Error=TRUE) THEN
            Stage:=INT#99;
        END_IF;

    3 :                             // Удаление каталога.
        DirRemove_instance(
            Execute:=TRUE,
            DirName:='Dir0',               // Имя каталога
```

```
            All    :=TRUE);                // Удалить файлы и подкаталоги.

        IF (DirRemove_instance.Done=TRUE) THEN
            Operating:=FALSE;               // Нормальное завершение
        END_IF;

        IF (DirRemove_instance.Error=TRUE) THEN
            Stage:=INT#99;                  // Завершение с ошибкой
        END_IF;

    99 :                            // Обработка после завершения с ошибкой.
        Operating:=FALSE;
    END_CASE;
END_IF;
```

FileRemove

Команда FileRemove удаляет указанный файл с карты памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileRemove	Удаление файла	FB		FileRemove_instance(Execute, FileName, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileName	Имя файла	Вход	Имя удаляемого файла	Макс. 66 байт (65 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	''

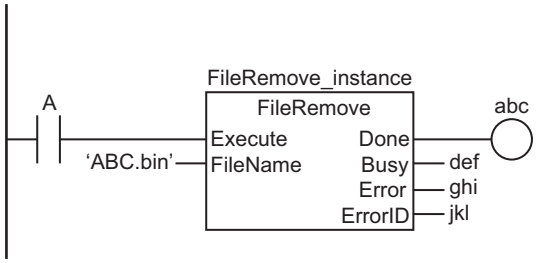
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileName																				OK

Функция

Команда FileRemove удаляет с карты памяти SD файл, указанный параметром *FileName* (имя файла).

Пример программы представлен на рисунке ниже. В данном случае удаляется файл с именем «ABC.bin».

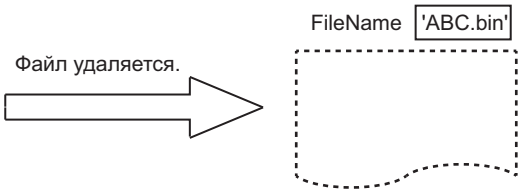
LD



ST

```
FileRemove_instance(A, 'ABC.bin', abc,
                    def, ghi, jkl);
```

Команда FileRemove удаляет с карты памяти SD файл, указанный параметром ***FileName***.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

Корневым каталогом в полном имени файла является каталог верхнего уровня карты памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- При переходе модуля ЦПУ в режим «Программирование» и при возникновении ошибки контроллера критического уровня любой открытый файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Если в то время, когда файл открыт, будет отключено питание нажатием кнопки питания карты памяти SD, файл поврежден не будет.
- Если в то время, когда файл открыт, карта памяти SD будет извлечена из гнезда без нажатия кнопки питания карты памяти SD, содержимое этого файла может быть повреждено. Обязательно выключайте питание карты памяти SD перед ее извлечением.
- Если в то время, когда файл открыт, будет отключено питание или карта памяти SD будет извлечена из гнезда, операции чтения и записи для этого файла будут невозможны, даже если карта памяти SD вновь будет вставлена.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Карта памяти SD защищена от записи.
  c) Файла с указанным именем *FileName* не существует.
  d) К файлу с указанным именем *FileName* в данный момент производится доступ.
  e) Файл с именем *FileName* уже существует и защищен от записи.
  f) Длина значения *FileName* превышает максимально допустимое количество байтов в имени файла.
  g) Выполняются одновременно пять или больше перечисленных ниже команд для работы с картой памяти SD, не имеющих параметра *FileID*: FileWriteVar, FileReadVar, FileCopy, DirCreate, FileRemove, DirRemove и FileRename.
  h) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

Ниже представлен пример использования команды для удаления с карты памяти SD файла с именем «ABC.bin».

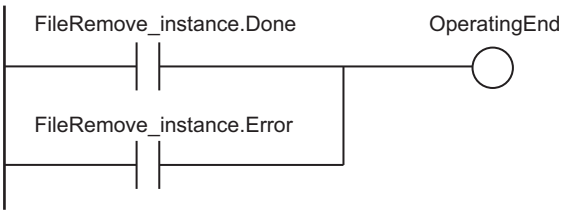
Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена.
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка

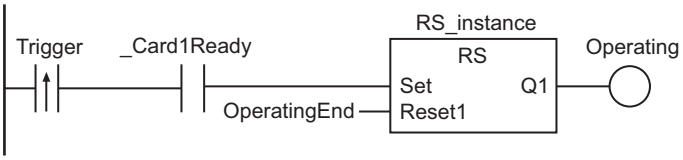
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	RS_instance	RS		
	FileRemove_instance	FileRemove		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

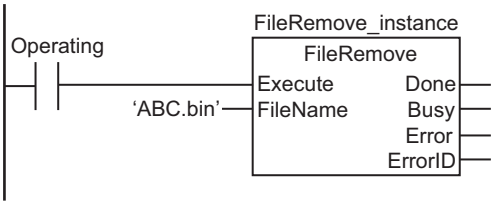
Проверка завершения выполнения команды FileRemove.



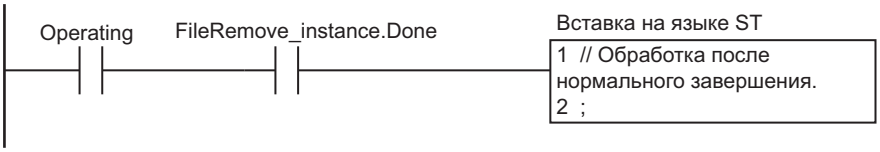
Прием условия выполнения.



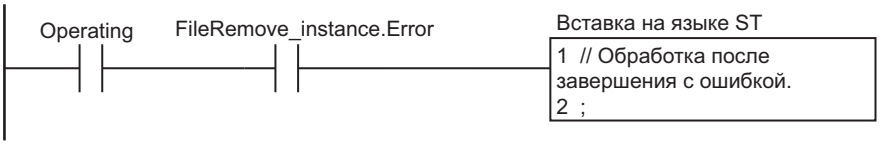
Выполнение команды FileRemove.



Обработка после нормального завершения.



Обработка после завершения с ошибкой



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение *Trigger* в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась.
	Operating	BOOL	ЛОЖЬ	Обработка
	FileRemove_instance	FileRemove		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация экземпляра.
IF (OperatingStart=TRUE) THEN
    FileRemove_instance(Execute:=FALSE);
    OperatingStart:=FALSE;
END_IF;

// Выполнение команды FileRemove.
IF (Operating=TRUE) THEN
    FileRemove_instance(
        Execute :=TRUE,
        FileName:='ABC.bin');   // Имя файла

    IF (FileRemove_instance.Done=TRUE) THEN
        Operating:=FALSE;       // Нормальное завершение
    END_IF;

    IF (FileRemove_instance.Error=TRUE) THEN
        Operating:=FALSE;       // Завершение с ошибкой
    END_IF;
END_IF;
```

FileRename

Команда FileRename изменяет имя указанного файла или каталога на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
FileRename	Изменение имени файла	FB	FileRename_instance FileRename Execute Done FileName Busy NewName Error OverWrite ErrorID	FileRename_instance(Execute, FileName, NewName, OverWrite, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
FileName	Исходное имя файла	Вход	Исходное имя файла	Макс. 66 байт (65 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	"
NewName	Новое имя файла		Новое имя файла			
OverWrite	Разрешение перезаписи		ИСТИНА: разрешить перезапись. ЛОЖЬ: запретить перезапись.	Зависит от типа данных.		ЛОЖЬ

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FileName																				ОК
NewName																				ОК
OverWrite	ОК																			

Функция

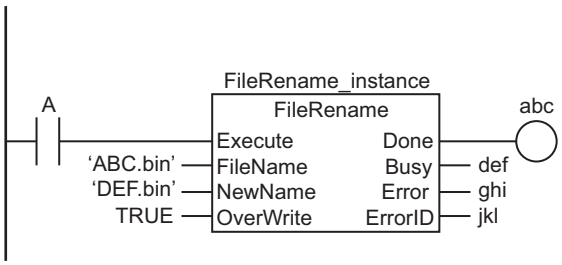
Команда FileRename меняет имя файла или каталога, указанное параметром *FileName* (исходное имя файла), на имя *NewName* (новое имя файла) на карте памяти SD.

Если файл или каталог с именем *NewName* уже существует на карте памяти SD, выполняется одна из следующих операций в зависимости от значения параметра *OverWrite* (разрешение перезаписи).

Значение *OverWrite*	Описание
ИСТИНА (разрешить перезапись.)	Существующий файл или каталог будет перезаписан.
ЛОЖЬ (запретить перезапись.)	Файл или каталог не перезаписывается, и возникает ошибка.

Пример программы представлен на рисунке ниже. В данном примере имя файла «ABC.bin» меняется на «DEF.bin».

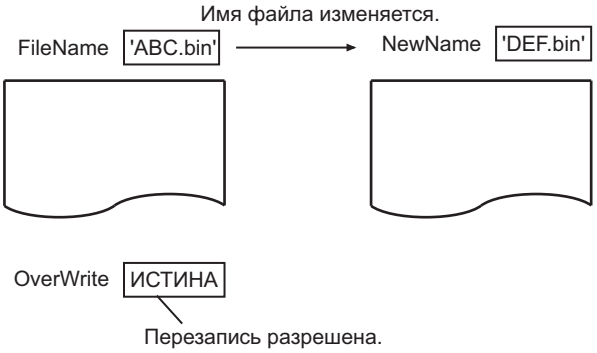
LD



ST

```
FileRename_instance(A, 'ABC.bin', 'DEF.bin',
                    TRUE, abc, def, ghi, jkl);
```

Команда FileRename меняет имя расположенного на карте памяти SD файла с именем ***FileName*** (исходное имя файла) на имя ***NewName*** (новое имя файла). Если файл уже существует, он перезаписывается.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.

Имя	Значение	Тип данных	Описание
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

Корневым каталогом в полном имени файла является каталог верхнего уровня карты памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если в параметрах *FileName* и *NewName* указаны разные каталоги, файл перемещается в каталог, указанный в *NewName*.
- При переходе модуля ЦПУ в режим «Программирование» и при возникновении ошибки контроллера критического уровня любой открытый файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Если в то время, когда файл открыт, будет отключено питание нажатием кнопки питания карты памяти SD, файл поврежден не будет.
- Если в то время, когда файл открыт, карта памяти SD будет извлечена из гнезда без нажатия кнопки питания карты памяти SD, содержимое этого файла может быть повреждено. Обязательно выключайте питание карты памяти SD перед ее извлечением.
- Если в то время, когда файл открыт, будет отключено питание или карта памяти SD будет извлечена из гнезда, операции чтения и записи для этого файла будут невозможны, даже если карта памяти SD вновь будет вставлена.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Карта памяти SD защищена от записи.
  c) Файла или каталога с указанным именем *FileName* не существует.
  d) Значение *FileName* или *NewName* не является допустимым именем файла или каталога.
  e) К файлу с указанным именем *FileName* в данный момент производится доступ.
  f) В каталоге, который указан для *FileName*, есть подкаталог, и значение *OverWrite* равно ИСТИНА.

g) Файл с именем *NewName* уже существует, и значение *OverWrite* равно ЛОЖЬ.
h) Файл с именем *NewName* уже существует и защищен от записи, а значение *OverWrite* = ИСТИНА.
i) Длина значения *NewName* превышает максимально допустимое количество байтов в имени файла или имени каталога.
j) Превышено максимально допустимое количество каталогов.
k) Выполняются одновременно пять или больше перечисленных ниже команд для работы с картой памяти SD, не имеющих параметра *FileID*: FileWriteVar, FileReadVar, FileCopy, DirCreate, FileRemove, DirRemove и FileRename.
l) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

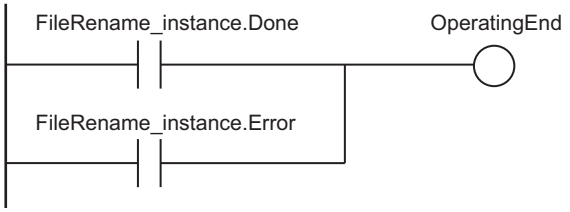
Ниже представлен пример использования команды для изменения имени файла на карте памяти SD с «ABC.bin» на «DEF.bin».

Программа на языке LD

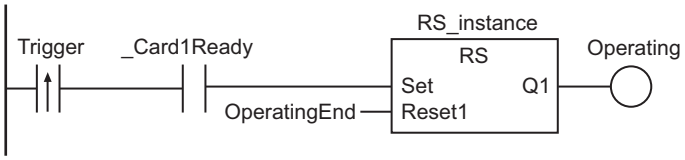
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OperatingEnd	BOOL	ЛОЖЬ	Обработка завершена.
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	Operating	BOOL	ЛОЖЬ	Обработка
	RS_instance	RS		
	FileRename_instance	FileRename		

Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

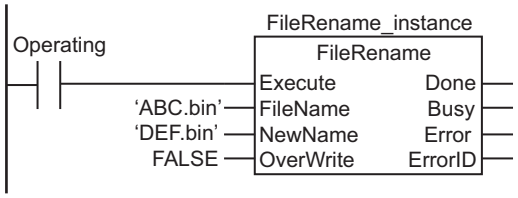
Проверка завершения выполнения команды FileRename.



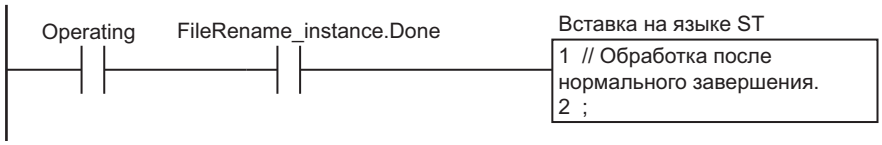
Прием условия выполнения.



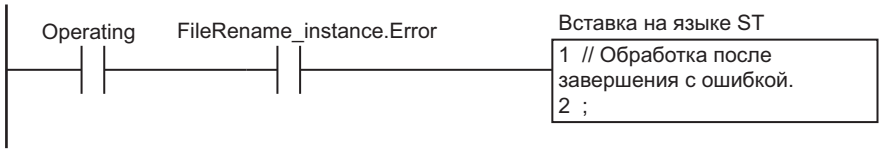
Выполнение команды FileRename.



Обработка после нормального завершения.



Обработка после завершения с ошибкой



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	Trigger	BOOL	ЛОЖЬ	Условие выполнения
	LastTrigger	BOOL	ЛОЖЬ	Значение *Trigger* в предыдущем цикле выполнения задачи
	OperatingStart	BOOL	ЛОЖЬ	Обработка началась.
	Operating	BOOL	ЛОЖЬ	Обработка
	FileRename_instance	FileRename		

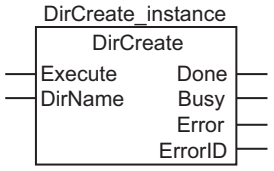
Внешние переменные	Переменная	Тип данных	Комментарий
	_Card1Ready	BOOL	Флаг готовности карты памяти SD

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) AND (_Card1Ready=TRUE) ) THEN
    OperatingStart:=TRUE;
    Operating     :=TRUE;
END_IF;
LastTrigger:=Trigger;

// Инициализация экземпляра.
IF (OperatingStart=TRUE) THEN
    FileRename_instance(Execute:=FALSE);
    OperatingStart:=FALSE;
END_IF;
```

```
// Выполнение команды FileRename.
IF (Operating=TRUE) THEN
    FileRename_instance(
        Execute  :=TRUE,
        FileName :='ABC.bin',    // Исходное имя файла
        NewName  :='DEF.bin',    // Новое имя файла
        OverWrite:=FALSE);       // Запретить перезапись.

    IF (FileRename_instance.Done=TRUE) THEN
        Operating:=FALSE;        // Нормальное завершение
    END_IF;

    IF (FileRename_instance.Error=TRUE) THEN
        Operating:=FALSE;        // Завершение с ошибкой
    END_IF;
END_IF;
```

DirCreate

Команда DirCreate создает каталог с указанным именем на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DirCreate	Создание каталога	FB		DirCreate_instance(Execute, DirName, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DirName	Создаваемый каталог	Вход	Имя создаваемого каталога	Макс. 66 байт (65 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	''

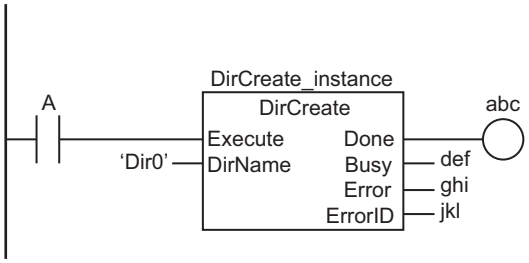
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DirName																				OK

Функция

Команда DirCreate создает на карте памяти SD каталог с именем *DirName* (создаваемый каталог).

Пример программы представлен на рисунке ниже. В данном случае создается каталог с именем «Dir0».

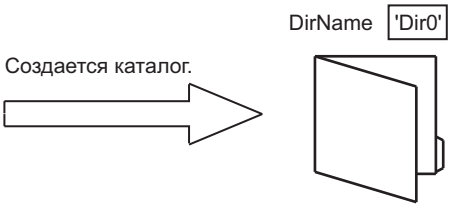
LD



ST

```
DirCreate_instance(A, 'Dir0', abc,
                   def, ghi, jkl);
```

Команда DirCreate создает на карте памяти SD каталог с именем, указанным параметром ***DirName***.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

Корневым каталогом в полном имени файла является каталог верхнего уровня карты памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- При переходе модуля ЦПУ в режим «Программирование» и при возникновении ошибки контроллера критического уровня любой открытый файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Если в то время, когда файл открыт, будет отключено питание нажатием кнопки питания карты памяти SD, файл поврежден не будет.
- Если в то время, когда файл открыт, карта памяти SD будет извлечена из гнезда без нажатия кнопки питания карты памяти SD, содержимое этого файла может быть повреждено. Обязательно выключайте питание карты памяти SD перед ее извлечением.
- Если в то время, когда файл открыт, будет отключено питание или карта памяти SD будет извлечена из гнезда, операции чтения и записи для этого файла будут невозможны, даже если карта памяти SD вновь будет вставлена.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Карта памяти SD защищена от записи.
  c) На карте памяти SD недостаточно свободного места.
  d) К файлу с указанным именем *FileName* в данный момент производится доступ.
  e) Превышено максимально допустимое количество каталогов.
  f) Выполняются одновременно пять или больше перечисленных ниже команд для работы с картой памяти SD, не имеющих параметра *FileID*: FileWriteVar, FileReadVar, FileCopy, DirCreate, FileRemove, DirRemove и FileRename.
  g) Каталог, указанный параметром *DirName*, уже существует.
  h) Значение *DirName* не является допустимым именем каталога.
  i) Длина значения *DirName* превышает максимально допустимое количество байтов в имени каталога.
  j) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

См. *Пример программы* на стр. 2-1597 для команды FileCopy.

DirRemove

Команда DirRemove удаляет указанный каталог с карты памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
DirRemove	Удаление каталога	FB	DirRemove_instance DirRemove Execute Done DirName Busy All Error ErrorID	DirRemove_instance(Execute, DirName, All, Done, Busy, Error, ErrorID);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DirName	Удаляемый каталог	Вход	Удаляемый каталог	Макс. 66 байт (65 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	"
All	Выбор объектов		Указывает, следует ли удалять файлы и подкаталоги внутри указанного каталога ИСТИНА: удалить файлы и подкаталоги. ЛОЖЬ: не удалять.	Зависит от типа данных.		ЛОЖЬ

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DirName																				OK
All	OK																			

Функция

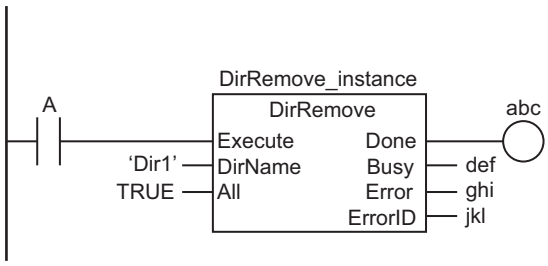
Команда DirRemove удаляет с карты памяти SD каталог, указанный параметром *DirName* (удаляемый каталог).

Если в указанном каталоге имеются файлы или подкаталоги, выполняется одна из следующих операций в зависимости от значения параметра *All* (выбор объектов).

Значение *All*	Описание
ИСТИНА	Вместе с указанным каталогом удаляются все файлы и подкаталоги.
ЛОЖЬ	Указанный каталог не удаляется, и возникает ошибка.

Пример программы представлен на рисунке ниже. В данном случае удаляется каталог с именем «Dir1».

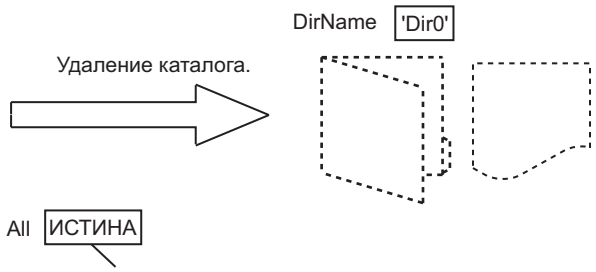
LD



ST

```
DirRemove_instance(A, 'Dir1', TRUE, abc,
                   def, ghi, jkl);
```

Команда DirRemove удаляет с карты памяти SD каталог с именем, указанным в ***DirName***.
Файлы и подкаталоги внутри указанного каталога также удаляются.



Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.

Имя	Значение	Тип данных	Описание
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

Корневым каталогом в полном имени файла является каталог верхнего уровня карты памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- При переходе модуля ЦПУ в режим «Программирование» и при возникновении ошибки контроллера критического уровня любой открытый файл будет закрыт системой. Любые выполнявшиеся в этот момент операции чтения или записи продолжат выполняться и будут выполнены до конца.
- Если в то время, когда файл открыт, будет отключено питание нажатием кнопки питания карты памяти SD, файл поврежден не будет.
- Если в то время, когда файл открыт, карта памяти SD будет извлечена из гнезда без нажатия кнопки питания карты памяти SD, содержимое этого файла может быть повреждено. Обязательно выключайте питание карты памяти SD перед ее извлечением.
- Если в то время, когда файл открыт, будет отключено питание или карта памяти SD будет извлечена из гнезда, операции чтения и записи для этого файла будут невозможны, даже если карта памяти SD вновь будет вставлена.
- Если каталог, указанный параметром *DirName*, защищен от записи, возникает ошибка и каталог не удаляется. Однако все файлы или каталоги внутри этого каталога, которые не защищены от записи, будут удалены.
- Не следует производить одновременно несколько разных операций доступа к одному и тому же файлу. Предусматривайте в программе пользователя эксклюзивное управление для команд для работы с картой памяти SD.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Карта памяти SD защищена от записи.
  c) Значение *All* = ИСТИНА, при этом к каталогу, который указан параметром *DirName*, в данный момент обращается другая команда.
  d) Значение *All* = ЛОЖЬ, при этом каталог, указанный параметром *DirName*, содержит файлы или подкаталоги.

e) Каталог, указанный параметром *DirName*, защищен от записи.
f) Каталог, указанный параметром *DirName*, содержит файлы или каталоги, защищенные от записи.
g) Каталога с указанным именем *DirName* не существует.
h) Длина значения *DirName* превышает максимально допустимое количество байтов в имени каталога.
i) Выполняются одновременно пять или больше перечисленных ниже команд для работы с картой памяти SD, не имеющих параметра *FileID*: FileWriteVar, FileReadVar, FileCopy, DirCreate, FileRemove, DirRemove и FileRename.
j) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

См. *Пример программы* на стр. 2-1597 для команды FileCopy.

BackupToMemoryCard

Команда BackupToMemoryCard создает резервную копию данных на карте памяти SD.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
BackupToMemoryCard	Резервное копирование на карту памяти SD	FB	BackupToMemoryCard_instance BackupToMemoryCard Execute, DirName, Cancel, Option Done, Busy, Error, Canceled, ErrorID	BackupToMemoryCard _instance(Execute, DirName, Cancel, Option, Done, Busy, Error, Canceled, ErrorID);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DirName	Целевой каталог	Вход	Имя каталога для хранения резервной копии	Макс. 64 байт (63 однобайтовых буквенно-цифровых символов + последний символ NULL)	---	"
Cancel	Отмена		Отмена резервного копирования ИСТИНА: отменить ЛОЖЬ: не отменять	Зависит от типа данных.		ЛОЖЬ
Option	Для будущего функционального расширения		Эта переменная предусмотрена для применения в будущем. Подключать параметр к ней не требуется.	---		---
Canceled	Отмена завершена	Выход	Флаг, указывающий, была ли завершена отмена. ИСТИНА: отмена завершена ЛОЖЬ: не удалось отменить	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DirName																				ОК
Cancel	ОК																			
Option	Для будущего функционального расширения. Подключать параметр к ней не требуется.																			
Canceled	ОК																			

Функция

Команда BackupToMemoryCard создает резервную копию данных на карте памяти SD.
Операция резервного копирования, выполняемая этой командой, полностью аналогична той, что выполняется с помощью DIP-переключателя на передней панели модуля ЦПУ, системной переменной _Card1BkupCmd или функции резервного копирования на карту памяти SD, предусмотренной в окне SD Memory Card Window (Карта памяти SD) в Sysmac Studio.

Имя каталога для хранения резервной копии данных указывается с помощью параметра *DirName*.
Если *DirName* содержит значение '' (т. е. текстовую строку длиной 0 символов), резервная копия сохраняется в корневом каталоге карты памяти SD.
Параметр *DirName* можно опустить. Если параметр *DirName* опущен, данные сохраняются следующим образом:

Выполнение команды	Каталог, в котором хранится копия
1-е выполнение	Корневой каталог
2-е и последующие выполнения	Ранее указанный каталог

Если каталога с указанным именем *DirName* на карте памяти SD не существует, создается новый каталог и резервная копия сохраняется в него.
Если в каталоге с указанным именем *DirName* уже есть файл с таким именем, как у файла резервной копии, файл резервной копии перезаписывается.

Если во время операции резервного копирования значение входа *Cancel* меняется на ИСТИНА, операция резервного копирования отменяется.
Если операция резервного копирования отменяется, файл резервной копии не создается.

Если в каталоге с указанным именем *DirName* уже есть файл резервной копии, файл резервной копии не перезаписывается и остается без изменений.
Можно отменить только операцию резервного копирования, которая выполняется для того же экземпляра функционального блока.
После завершения отмены операции значение переменной *Canceled* меняется на ИСТИНА.
Если вход *Cancel* не будет своевременно переведен в состояние ИСТИНА, изменение его значения для отмены операции может быть не распознано вовремя и операция резервного копирования может быть продолжена до полного завершения. Если изменение значения для отмены операции не произойдет вовремя, выход *Canceled* будет находиться в состоянии ЛОЖЬ, а выход *Done* — в состоянии ИСТИНА.

Если вход *Cancel* находится в состоянии ИСТИНА, операция резервного копирования не выполняется, даже если состояние входа *Execute* меняется на ИСТИНА.

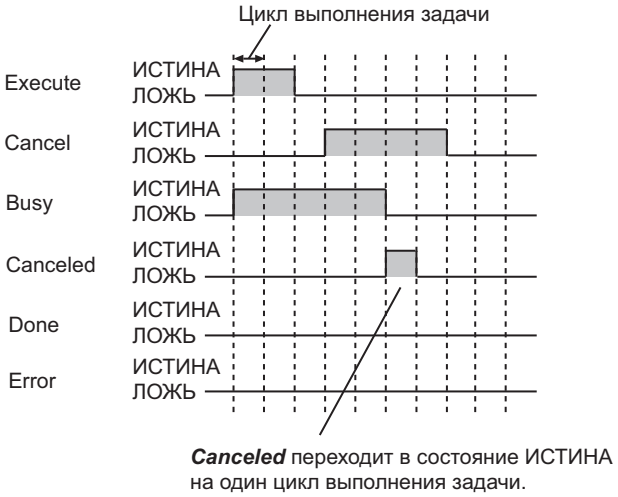
Вход *Option* предусмотрен для будущего функционального расширения. Не подключайте к нему какой-либо параметр.

Временная диаграмма отмены операции

Ниже представлены временные диаграммы изменения переменных команды в случае отмены операции.

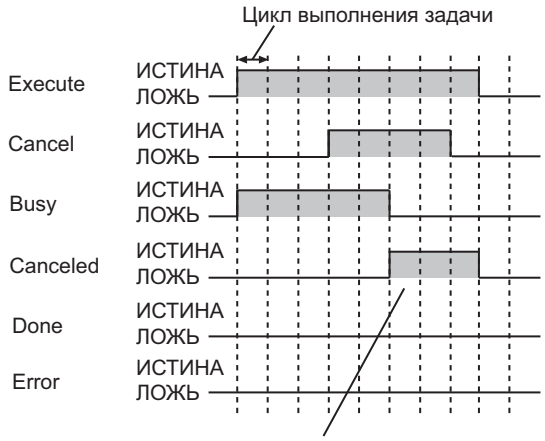
● Перевод входа *Execute* в состояние ЛОЖЬ до перехода выхода *Canceled* в состояние ИСТИНА для успешной отмены операции

- Операция резервного копирования запускается, когда вход *Execute* переходит в состояние ИСТИНА. Состояние выхода *Busy* меняется на ИСТИНА.
- Операция резервного копирования отменяется, когда вход *Cancel* переходит в состояние ИСТИНА.
- После завершения отмены операции выход *Busy* переходит в состояние ЛОЖЬ, а выход *Canceled* — в состояние ИСТИНА.
- Вход *Execute* переводится в состояние ЛОЖЬ до того, как выход *Canceled* переходит в состояние ИСТИНА.
- Выход *Canceled* возвращается в состояние ЛОЖЬ в следующем цикле выполнения задачи.
- Поскольку отмена успешно выполнена, выход *Done* остается в состоянии ЛОЖЬ.



● Перевод входа *Execute* в состояние ЛОЖЬ после перехода выхода *Canceled* в состояние ИСТИНА для успешной отмены операции

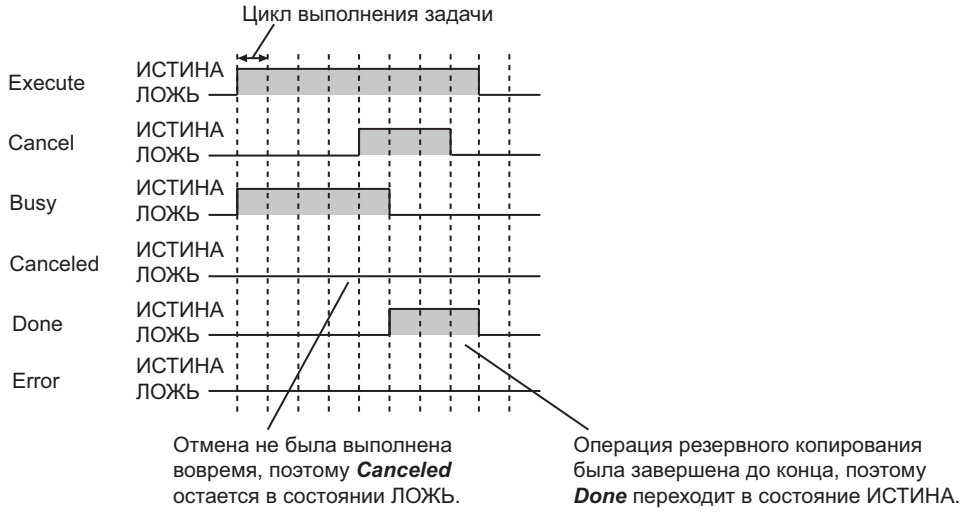
- Операция резервного копирования запускается, когда вход *Execute* переходит в состояние ИСТИНА. Состояние выхода *Busy* меняется на ИСТИНА.
- Операция резервного копирования отменяется, когда вход *Cancel* переходит в состояние ИСТИНА.
- После завершения отмены операции выход *Busy* переходит в состояние ЛОЖЬ, а выход *Canceled* — в состояние ИСТИНА.
- Вход *Execute* переводится в состояние ЛОЖЬ после того, как выход *Canceled* перешел в состояние ИСТИНА.
- Выход *Canceled* будет оставаться в состоянии ИСТИНА до тех пор, пока вход *Execute* не перейдет в состояние ЛОЖЬ.
- Поскольку отмена успешно выполнена, выход *Done* остается в состоянии ЛОЖЬ.



Canceled будет оставаться в состоянии ИСТИНА, пока ***Execute*** не поменяется на ЛОЖЬ.

● Если отмена не производится вовремя

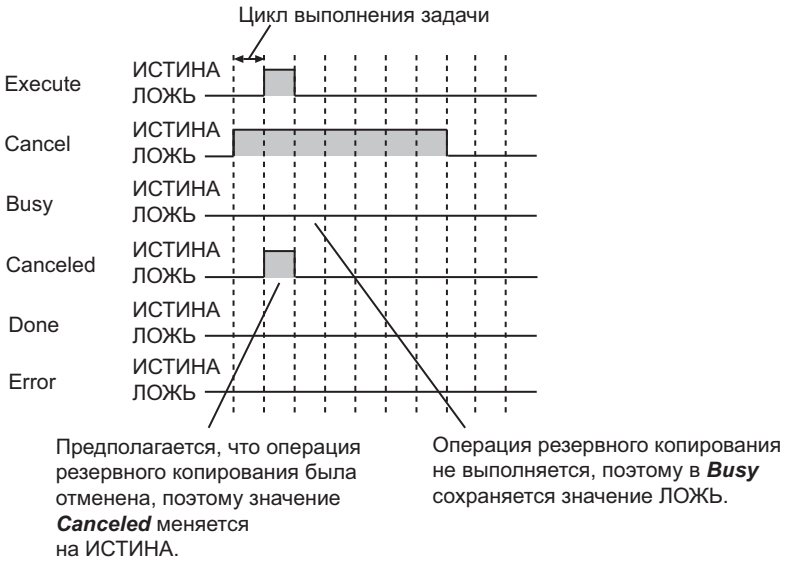
- Операция резервного копирования запускается, когда вход *Execute* переходит в состояние ИСТИНА. Состояние выхода *Busy* меняется на ИСТИНА.
- Состояние входа *Cancel* меняется на ИСТИНА. Поскольку изменение значения для отмены операции произошло слишком поздно, операция резервного копирования продолжается.
- После завершения операции резервного копирования выход *Busy* переходит в состояние ЛОЖЬ.
- Так как операция резервного копирования была продолжена и выполнена до конца, выход *Done* переходит в состояние ИСТИНА.
- Отмена операции не была инициирована вовремя, поэтому выход *Canceled* остается в состоянии ЛОЖЬ.



● Вход *Execute* переводится в состояние ИСТИНА, когда вход *Cancel* находится в состоянии ИСТИНА

- Вход *Cancel* переводится в состояние ИСТИНА.
- Операция резервного копирования не запускается, даже если вход *Execute* переходит в состояние ИСТИНА. Поэтому выход *Busy* остается в состоянии ЛОЖЬ.
- Предполагается, что операция резервного копирования была отменена, поэтому выход *Canceled* переходит в состояние ИСТИНА.

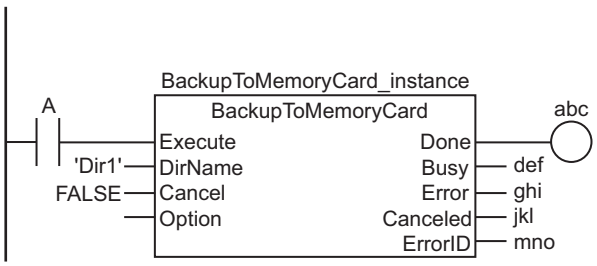
- Если вход *Execute* перейдет в состояние ЛОЖЬ, выход *Canceled* также перейдет в состояние ЛОЖЬ.



Пример записи

Пример программы представлен на рисунке ниже. Файл резервной копии сохраняется в каталог с именем «Dir1».

LD



ST

```
BackupToMemoryCard_instance(A, 'Dir1', FALSE,
                            , abc, def, ghi, jkl,
                            mno);
```

Связанные системные переменные

Имя	Значение	Тип данных	Описание
_Card1Ready	Флаг готовности карты памяти SD	BOOL	Этот флаг указывает, возможен ли доступ к карте памяти SD с помощью команд программы и команд связи. *1 ИСТИНА: использование возможно. ЛОЖЬ: использование невозможно.
_Card1Protect	Флаг защиты от записи карты памяти SD	BOOL	Этот флаг указывает, защищена ли карта памяти SD от записи, когда она вставлена и готова к использованию. ИСТИНА: запись запрещена. ЛОЖЬ: запись не запрещена.
_Card1Err	Флаг ошибки карты памяти SD	BOOL	Этот флаг указывает, что установлена карта памяти SD, которую нельзя использовать, или что произошла ошибка форматирования. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.

Имя	Значение	Тип данных	Описание
_Card1Access	Флаг доступа к карте памяти SD	BOOL	Этот флаг указывает, выполняется ли в данный момент доступ к карте памяти SD. ИСТИНА: производится доступ. ЛОЖЬ: доступ не производится.
_Card1Deteriorated	Флаг предупреждения о сроке службы карты памяти SD	BOOL	Этот флаг указывает, не обнаружено ли окончание срока службы карты памяти SD. ИСТИНА: обнаружено окончание срока службы. ЛОЖЬ: не обнаружено.
_Card1PowerFail	Флаг прерывания питания карты памяти SD	BOOL	Этот флаг указывает, не произошла ли ошибка при выполнении операции из-за того, что во время доступа было прервано питание.*2 Этот флаг не сбрасывается автоматически. ИСТИНА: ошибка. ЛОЖЬ: ошибок нет.
_BackupBusy	Флаг занятости функции резервного копирования	BOOL	Этот флаг указывает, не выполняется ли в данный момент операция резервного копирования, восстановления или сверки данных. ИСТИНА: выполняется операция резервного копирования, восстановления или сравнения. ЛОЖЬ: операция резервного копирования, восстановления или сравнения не выполняется в данный момент.

*1. Обязательное предварительное условие: карта памяти SD должна быть вставлена в соответствующий слоти нормально смонтирована.

*2. Подразумевается доступ к карте памяти SD.

Дополнительная информация

- Дополнительные сведения о функциях резервного копирования см. в документе *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.
- Корневым каталогом в полном имени файла является каталог верхнего уровня карты памяти SD.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Выполнение этой команды будет продолжаться до завершения, даже если значение *Execute* поменяется на ЛОЖЬ или время выполнения превысит период выполнения задачи. По завершении выполнения значение *Done* поменяется на ИСТИНА. Эту переменную можно использовать в качестве флага нормального завершения обработки.
- Временную диаграмму для переменных *Execute*, *Done*, *Busy* и *Error* см. в разделе *Использование данной главы* на стр. 2-3.
- Если в то время, когда файл открыт, карта памяти SD будет извлечена из гнезда без нажатия кнопки питания карты памяти SD, содержимое этого файла в некоторых случаях может быть повреждено. Обязательно выключайте питание карты памяти SD перед ее извлечением.
- Эту команду можно выполнять для создания резервной копии данных даже в том случае, когда резервное копирование данных на карту памяти SD запрещено. Ошибки при этом не произойдет.
- Значения указанных ниже системных переменных, связанных с резервным копированием, при выполнении этой команды не изменяются.
  a) Управление резервным копированием на карту памяти SD: _CardBkupCmd
  b) Состояние резервного копирования на карту памяти SD: _Card1BkupSta

- Во время выполнения этой команды не следует производить чтение или запись файлов, связанных с резервным копированием. Чтение данных из файла во время записи в него может привести к непредсказуемым последствиям.
- Даже если во время выполнения этой команды поменяется режим работы модуля ЦПУ, операция резервного копирования продолжится. Если контроллер будет переведен из режима «Выполнение» в режим «Программирование», а затем возвращен в режим «Выполнение», выход *Busy* останется в состоянии ЛОЖЬ, даже если операция резервного копирования все еще выполняется. Если в данной ситуации операция резервного копирования будет отменена, выход *Canceled* перейдет в состояние ИСТИНА.
- В указанных ниже случаях произойдет ошибка. Значение *Error* поменяется на ИСТИНА.
  a) Карта памяти SD не находится в состоянии, в котором возможно ее использование.
  b) Карта памяти SD защищена от записи.
  c) На карте памяти SD недостаточно свободного места.
  d) Превышено максимально допустимое количество файлов или каталогов.
  e) Уже существует файл с именем, которое указано в *DirName*.
  f) Значение *DirName* не является допустимым именем каталога.
  g) Уже выполняется другая операция резервного копирования.
  h) Не удалось выполнить резервное копирование.
  i) Во время доступа к карте памяти SD возникает ошибка, которая приводит к сбою доступа.

Пример программы

В данном примере команда BackupToMemoryCard создает резервную копию данных на карте памяти SD каждый день ровно в полночь.
Файлы резервных копий хранятся в каталогах с именем /Backup/гггг-мм-дд на карте памяти SD.
Имя каталога содержит дату создания резервной копии.
"гггг" — год, "мм" — месяц, а "дд" — день месяца.

Описание сенсорной панели

В этом примере предполагается, что сенсорная панель подключена к контроллеру.
Ниже перечислены индикаторы, предусмотренные на сенсорной панели.

Наименование индикатора	Описание
Индикатор нормального завершения резервного копирования	Загорается после нормального завершения операции резервного копирования.
Индикатор отмены резервного копирования	Загорается после успешной отмены операции резервного копирования.
Индикатор завершения резервного копирования с ошибкой	Загорается, если операция резервного копирования завершается ошибкой.
Индикатор предупреждения о сроке службы карты памяти SD	Горит, если эксплуатационный ресурс карты памяти SD истек.
Индикатор прерывания питания карты памяти SD	Горит, если во время операции резервного копирования было отключено питание карты памяти SD.

На сенсорной панели также предусмотрены указанные ниже кнопки.

Наименование кнопки	Действие при нажатии кнопки
Кнопка выключения индикаторов	Выключает индикатор нормального завершения резервного копирования, индикатор отмены резервного копирования, индикатор завершения резервного копирования с ошибкой и индикатор прерывания питания карты памяти SD.
Кнопка отмены.	Отменяет резервное копирование.

Определения глобальных переменных

● Глобальные переменные

Переменная	Тип данных	Начальное значение	Комментарий
PTOut_Warning_SDLife	BOOL	ЛОЖЬ	Выходной сигнал на индикатор предупреждения о сроке службы карты памяти SD
PTOut_Warning_PwrFail_onBackup	BOOL	ЛОЖЬ	Выходной сигнал на индикатор прерывания питания карты памяти SD
PTOut_Done	BOOL	ЛОЖЬ	Выходной сигнал на индикатор нормального завершения резервного копирования
PTOut_Cancel	BOOL	ЛОЖЬ	Выходной сигнал на индикатор отмены резервного копирования
PTOut_Error	BOOL	ЛОЖЬ	Выходной сигнал на индикатор завершения резервного копирования с ошибкой
PTIn_Check_Backup	BOOL	ЛОЖЬ	Входной сигнал от кнопки выключения индикаторов
PTIn_Cancel	BOOL	ЛОЖЬ	Входной сигнал от кнопки отмены

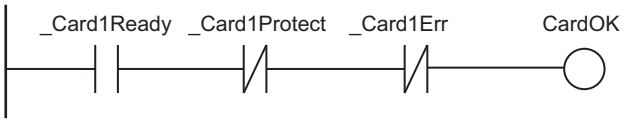
Программа на языке LD

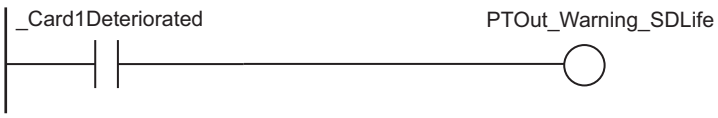
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	CardOK	BOOL	ЛОЖЬ	Флаг нормального состояния карты памяти SD
	Backup_inst	BackupToMemoryCard		Экземпляр команды BackupToMemoryCard
	PreviousDay	USINT	0	Дата предыдущего цикла выполнения задачи
	CurrentDT	DATE_AND_TIME	ST#1970-01-01-00:00:00.000000000	Текущие дата и время
	Current_sDt	_sDT	(Year:=0, Month:=0, Day:=0, Hour:=0, Min:=0, Sec:=0, NSec:=0)	Текущие дата и время в виде отдельных значений года, месяца, дня, часа, минут, секунд и наносекунд.
	BackupCondition	BOOL	ЛОЖЬ	Флаг наступления условия резервного копирования
	tmpString	STRING[256]	''	Временная текстовая строка, которая используется для создания имени каталога
	tmpString2	STRING[256]	''	Временная текстовая строка, которая используется для создания имени каталога

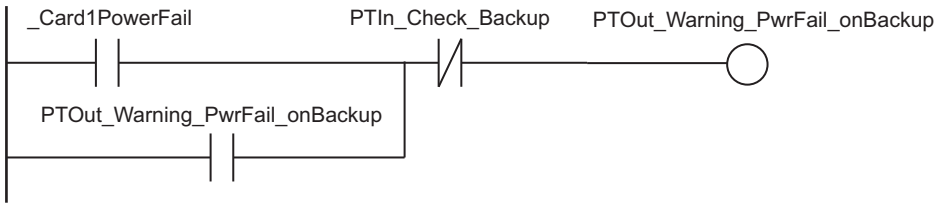
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	BackupPath	STRING[64]	''	Имя каталога
	Cancel	BOOL	ЛОЖЬ	Флаг наступления условия отмены операции.

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_Card1Ready	BOOL	☑	Флаг готовности карты памяти SD
	_Card1Protect	BOOL	☑	Флаг защиты от записи карты памяти SD
	_Card1Err	BOOL	☑	Флаг ошибки карты памяти SD
	_Card1Deteriorated	BOOL	☑	Флаг предупреждения о сроке службы карты памяти SD
	_Card1PowerFail	BOOL	☐	Флаг прерывания питания карты памяти SD
	_BackupBusy	BOOL	☑	Флаг занятости функции резервного копирования
	PTOut_Warning_SDLife	BOOL	☐	Выходной сигнал на индикатор предупреждения о сроке службы карты памяти SD
	PTOut_Warning_PwrFail_onBackup	BOOL	☐	Выходной сигнал на индикатор прерывания питания карты памяти SD
	PTOut_Done	BOOL	☐	Выходной сигнал на индикатор нормального завершения резервного копирования
	PTOut_Cancel	BOOL	☐	Выходной сигнал на индикатор отмены резервного копирования
	PTOut_Error	BOOL	☐	Выходной сигнал на индикатор завершения резервного копирования с ошибкой
	PTIn_Check_Backup	BOOL	☐	Входной сигнал от кнопки выключения индикаторов
	PTIn_Cancel	BOOL	☐	Входной сигнал от кнопки отмены

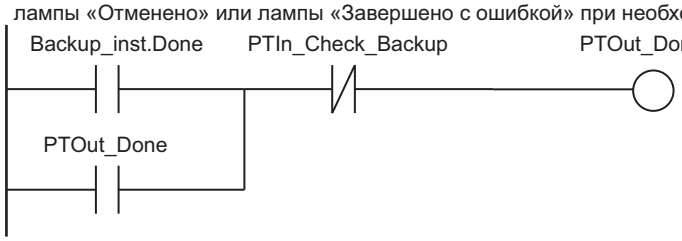
Проверка состояния карты памяти SD.

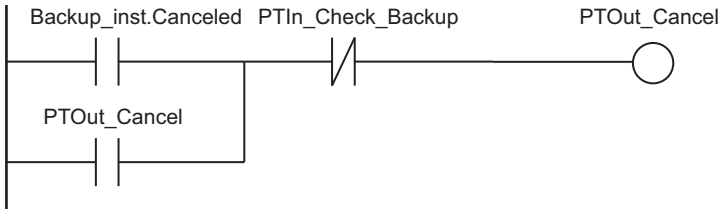


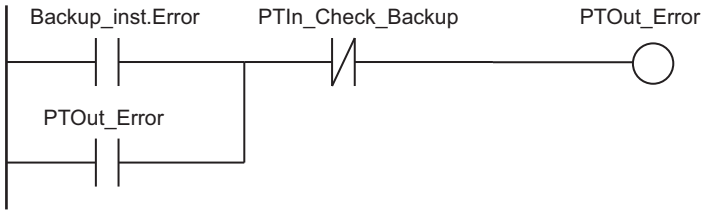




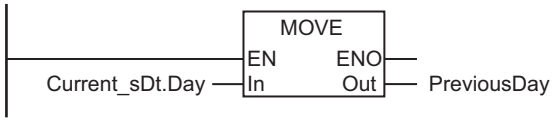
Включение лампы «Нормальное завершение резервного копирования», лампы «Отменено» или лампы «Завершено с ошибкой» при необходимости.

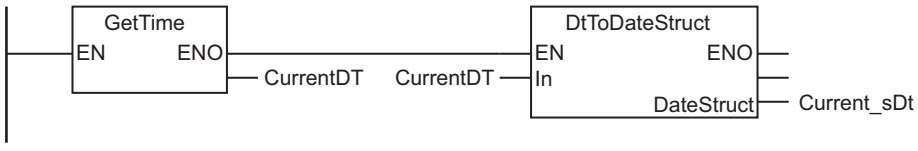


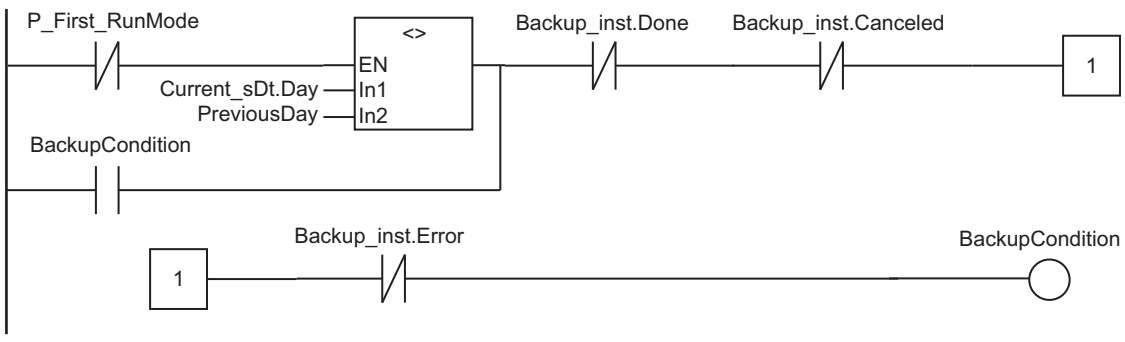




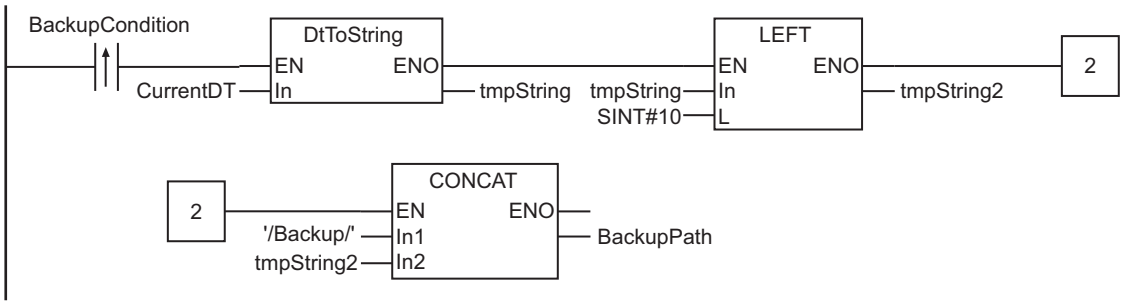
Проверка, изменилась ли дата.



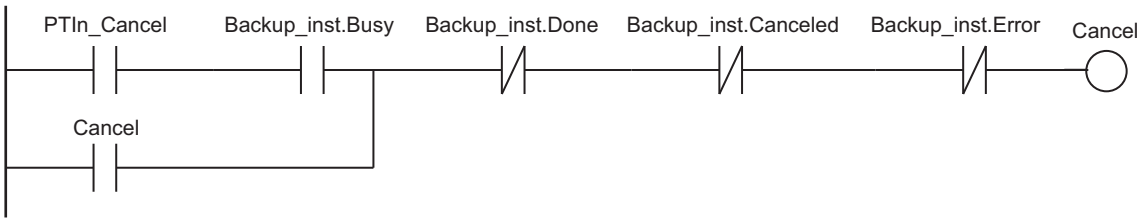




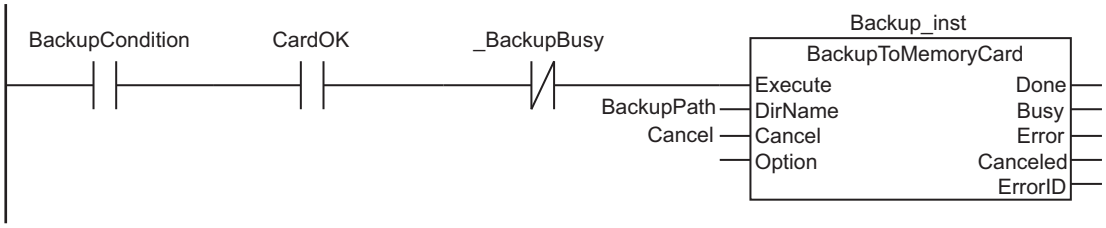
Создание имени каталога.



Обнаружение нажатия кнопки «Отмена».



Выполнение команды BackupToMemoryCard.



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	CardOK	BOOL	ЛОЖЬ	Флаг нормального состояния карты памяти SD
	Backup_inst	BackupToMemoryCard		Экземпляр команды BackupToMemoryCard
	PreviousDay	USINT	0	Дата предыдущего цикла выполнения задачи
	CurrentDT	DATE_AND_TIME	ST#1970-01-01-00:00:00.000000000	Текущие дата и время
	Current_sDt	_sDT	(Year:=0, Month:=0, Day:=0, Hour:=0, Min:=0, Sec:=0, NSec:=0)	Текущие дата и время в виде отдельных значений года, месяца, дня, часа, минут, секунд и наносекунд.
	BackupCondition	BOOL	ЛОЖЬ	Флаг наступления условия резервного копирования

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	tmpString	STRING[256]	''	Временная текстовая строка, которая используется для создания имени каталога
	tmpString2	STRING[256]	''	Временная текстовая строка, которая используется для создания имени каталога
	BackupPath	STRING[64]	''	Имя каталога
	Cancel	BOOL	ЛОЖЬ	Флаг наступления условия отмены операции
	RS1	RS		Экземпляр 1 команды RS (Удержание с приоритетом сброса)
	RS2	RS		Экземпляр 2 команды RS (Удержание с приоритетом сброса)
	RS3	RS		Экземпляр 3 команды RS (Удержание с приоритетом сброса)
	RS4	RS		Экземпляр 4 команды RS (Удержание с приоритетом сброса)
	RS5	RS		Экземпляр 5 команды RS (Удержание с приоритетом сброса)
	RS6	RS		Экземпляр 6 команды RS (Удержание с приоритетом сброса)

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	_Card1Ready	BOOL	☑	Флаг готовности карты памяти SD
	_Card1Protect	BOOL	☑	Флаг защиты от записи карты памяти SD
	_Card1Err	BOOL	☑	Флаг ошибки карты памяти SD
	_Card1Deteriorated	BOOL	☑	Флаг предупреждения о сроке службы карты памяти SD
	_Card1PowerFail	BOOL	☐	Флаг прерывания питания карты памяти SD
	_BackupBusy	BOOL	☑	Флаг занятости функции резервного копирования
	PTOut_Warning_SDLife	BOOL	☐	Выходной сигнал на индикатор предупреждения о сроке службы карты памяти SD
	PTOut_Warning_PwrFail_onBackup	BOOL	☐	Выходной сигнал на индикатор прерывания питания карты памяти SD
	PTOut_Done	BOOL	☐	Выходной сигнал на индикатор нормального завершения резервного копирования

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	PTOut_Cancel	BOOL	☐	Выходной сигнал на индикатор отмены резервного копирования
	PTOut_Error	BOOL	☐	Выходной сигнал на индикатор завершения резервного копирования с ошибкой
	PTIn_Check_Backup	BOOL	☐	Входной сигнал от кнопки выключения индикаторов
	PTIn_Cancel	BOOL	☐	Входной сигнал от кнопки отмены

```
// Проверка состояния карты памяти SD.
CardOK := _Card1Ready OR NOT(_Card1Protect) OR NOT(_Card1Err);
PTOut_Warning_SDCardLife := _Card1Deteriorated;
RS1(Set := _Card1PowerFail, Reset1 := PTIn_Check_Backup, Q1=>PTOut_Warning_PwrFail_
onBackup);

 // Включить индикатор нормального завершения резервного копирования, индикатор отм
ены или индикатор завершения с ошибкой по ситуации.
 RS2(Set    := Backup_inst.Done,
     Reset1 := PTIn_Check_Backup,
     Q1     => PTOut_Done);
 RS3(Set    := Backup_inst.Canceled,
     Reset1 := PTIn_Check_Backup,
     Q1     => PTOut_Cancel);
 RS4(Set    := Backup_inst.Error,
     Reset1 := PTIn_Check_Backup,
     Q1     => PTOut_Error);

 // Проверка, изменилась ли дата.
 PreviousDay := Current_sDT.Day;
 CurrentDT:=GetTime();
 DtToDateStruct(In := CurrentDT,DateStruct=>Current_sDT);
 RS5(Set    := ( NOT (P_First_RunMode) & (Current_sDT.Day<>PreviousDay),
     Reset1 := (Backup_inst.Done OR Backup_inst.Canceled OR Backup_inst.Error),
     Q1     => BackupCondition);

 // Создание имени каталога.
 IF(BackupCondition) THEN
    BackupPath := CONCAT('/Backup/', Left(In:= DtToString(CurrentDT), L:=SINT#10));
 END_IF;

 // Обнаружение нажатия кнопки «Отмена».
 RS6(Set    := (PTIn_Cancel &Backup_inst.Busy),
     Reset1 := (Backup_inst.Done OR Backup_inst.Canceled OR Backup_inst.Error),
     Q1     => Cancel);

// Выполнение команды BackupToMemoryCard.
Backup_inst(Execute := (BackupCondition & CardOK & NOT (_BackupBusy)),
```

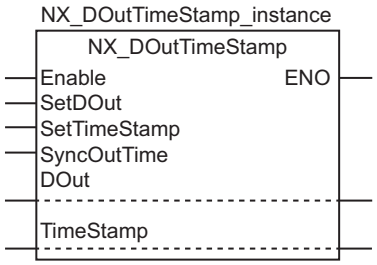
```
DirName := BackupPath,
Cancel  := Cancel);
```

Команды с использованием меток времени

Команда	Имя	Стр.
NX_DOutTimeStamp	Запись в дискретный выход с указанием метки времени	стр. 2-1636
NX_AryDOutTimeStamp	Запись в дискретный выход из массива с указанием метки времени	стр. 2-1643

NX_DOutTimeStamp

Команда NX_DOutTimeStamp записывает значение в выходной бит модуля дискретных выходов, поддерживающего обновление с использованием меток времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_DOutTimeStamp	Запись в дискретный выход с указанием метки времени	FB		NX_DOutTimeStamp_instance(Enable, SetDOut, SetTimeStamp, SyncOutTime, DOut, TimeStamp);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.06 или более поздней и Sysmac Studio версии 1.07 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Enable	Активация	Вход	ИСТИНА: выводится значение *SetDOut*. ЛОЖЬ: когда значение этой переменной меняется на ЛОЖЬ, выходное значение также меняется на ЛОЖЬ.	Зависит от типа данных.	---	ЛОЖЬ
SetDOut	Выходное значение		Выходное значение			
SetTimeStamp	Заданная метка времени		Время вывода значения		нс	0
SyncOutTime	Метка времени синхронного вывода		Переменная устройства *Time Stamp of Synchronous Output (Метка времени синхронного вывода)* интерфейсного модуля EtherCAT или модуля NX в стойке модуля ЦПУ			*1

	Значение	Вход/выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
DOut	Выходной бит модуля дискретных выходов	Вход-выход	Переменная устройства *Output Bit* ** модуля дискретных выходов, поддерживающего обновление с указанием метки времени	Зависит от типа данных.	---	---
TimeStamp	Метка времени		Переменная устройства *Output Bit* ** *Time Stamp* модуля дискретных выходов, поддерживающего обновление с указанием метки времени		нс	

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	ОК																			
SetDOut	ОК																			
SetTimeStamp									ОК											
SyncOutTime									ОК											
DOut	ОК																			
TimeStamp									ОК											

Функция

Когда значение переменной *Enable* = ИСТИНА, команда NX_DOutTimeStamp в заданное время записывает значение *SetDOut* (выходное значение) в выходной бит модуля дискретных выходов, поддерживающего обновление с указанием метки времени.
Если вход *Enable* переходит в состояние ЛОЖЬ, значение выходного бита меняется на ЛОЖЬ, начиная со следующего цикла выполнения задачи.
Разница между фактическим временем вывода значения и заданным временем не превышает ±1 мкс.

Значение *SyncOutTime* (метка времени синхронного вывода) определяется на основе информации часов в интерфейсном модуле EtherCAT или модуле NX, подключенном к шине NX модуля ЦПУ, к которому подключен модуль дискретных выходов, поддерживающий обновление с указанием метки времени.

Следует указывать переменную устройства *Time Stamp of Syncronous Output (Метка времени синхронного вывода)* интерфейсного модуля EtherCAT или модуля NX, подключенного к шине NX модуля ЦПУ, к которому подключен модуль дискретных выходов.

Однако в случае интерфейсного модуля EtherCAT к записям ввода-вывода необходимо добавлять 0x200A:02 (метка времени синхронного вывода).

В качестве параметра *DOut* (выходной бит модуля дискретных выходов) следует указывать переменную устройства *Output Bit ***, которая назначена выходному биту модуля дискретных выходов, поддерживающего обновление с указанием метки времени.

В качестве параметра *TimeStamp* следует указывать переменную устройства *Output Bit ** Time Stamp*, которая назначена метке времени выходного бита модуля дискретных выходов, поддерживающего обновление с указанием метки времени.

Указание времени вывода

Для указания времени вывода следует соблюдать приведенный ниже порядок действий.

1 Получите значение переменной устройства, которая назначена информации часов (служит в качестве опорного времени для бита модуля).

2 Вычислите разницу между полученной информацией часов и временем записи данных в выходной бит в наносекундах и добавьте ее к значению переменной устройства из шага 1.

3 Передайте результат сложения в переменную *SetTimeStamp* (заданная метка времени) команды NX_DOutTimeStamp.

Дополнительные сведения см. в разделе *Пример программы* на стр. 2-1639 для этой команды.

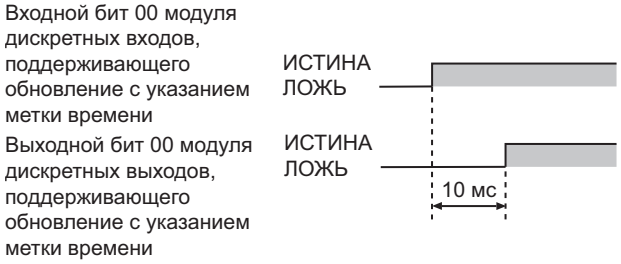
Меры предосторожности для обеспечения надлежащей эксплуатации

- Эту команду можно выполнить только для модуля дискретных выходов, который поддерживает обновление с использованием метки времени. Но даже если в системе нет ни одного подключенного модуля дискретных выходов, поддерживающего обновление с использованием метки времени, и будет выполнена эта команда, к возникновению ошибки это не приведет.
- Если произойдет ошибка связи по сети EtherCAT или будет превышен период выполнения задачи, запись может быть не выполнена в указанное время. В этом случае значение выводится в следующем цикле выполнения задачи или еще позже.
- Если какая-либо переменная устройства, используемая для этой команды, изменяется или читается любой другой командой или программой, необходимо предусмотреть эксклюзивное управление.
- Параметр *SyncOutTime* следует указывать с помощью переменной устройства *Time Stamp of Syncronous Output (Метка времени синхронного вывода)* интерфейсного модуля EtherCAT или модуля NX, подключенного к шине NX модуля ЦПУ, к которому подключен модуль дискретных выходов, поддерживающий обновление с указанием метки времени.
  Но даже если будет указана другая переменная, ошибки не произойдет.
- В качестве параметров *DOut* и *TimeStamp* следует указывать соответствующие переменные устройства модуля дискретных выходов, поддерживающего обновление с указанием метки времени. Но даже если будет указана другая переменная, ошибки не произойдет.

- В качестве параметров *DOut* и *TimeStamp* следует указывать соответствующие переменные устройства для того же канала того же модуля. Но даже если будет указана другая переменная, ошибки не произойдет.
- Если для переменной *TimeStamp* задано время в прошлом, значение *TimeStamp* становится равно 0.
  В этом случае состояние выхода модуля дискретных выходов, поддерживающего обновление с указанием метки времени, обновляется немедленно.
  Дополнительные сведения см. в документе *Серия NX, модули дискретных входов-выходов — Руководство пользователя (Cat. No. W521)*.

Пример программы

Рассмотрим пример программы для реализации следующего алгоритма: выходной бит 00 модуля дискретных выходов, поддерживающего обновление выходов с указанием метки времени, переходит в состояние ИСТИНА через 10 мс после того, как значение входного бита 00 модуля дискретных входов меняется на ИСТИНА.
Предполагается, что входной бит 00 остается в состоянии ИСТИНА дольше, чем длится цикл обновления входов-выходов на шине NX.
В данном примере переход входного бита 00 в состояние ИСТИНА играет роль входного сигнала запуска.
Однако если входной бит 00 вернется в состояние ЛОЖЬ раньше, чем закончится цикл обновления входов-выходов на шине NX, переход входного бита 00 в состояние ИСТИНА может быть не обнаружен. Для решения этой проблемы можно, например, использовать в качестве события запуска зарегистрированное время изменения входного бита 00.
В документе *Серия NX, модули дискретных входов-выходов — Руководство пользователя (Cat. No. W521)* подробно рассмотрен пример программы, которая включает определенный выход по истечении заданного промежутка времени после изменения состояния входа датчика.



Конфигурация сети

В таблице ниже перечислены модули, которые входят в конфигурацию сети. Ведомый терминал, состоящий из указанных ниже модулей, подключен к узлу EtherCAT с адресом 1. Модулям присвоены имена устройств, указанные в таблице.

Номер модуля	Номер модели	Модуль	Имя устройства
0	NX-ECC201	Интерфейсный модуль EtherCAT	E001
1	NX-ID3344	Модуль дискретных входов, поддерживающий обновление с использованием меток времени	N1
2	NX-OD2154	Модуль дискретных выходов, поддерживающий обновление с использованием меток времени	N2

Настройка рабочих параметров модуля

В таблице ниже приведены значения рабочих параметров модуля дискретных входов, поддерживающего обновление с использованием меток времени.

Параметр	Заданное значение	Значение
Метка времени (настройка события запуска): настройка события запуска для входного бита 00	ЛОЖЬ	Фронт для чтения зарегистрированного времени изменения входа: положительный фронт
Метка времени (настройка режима): настройка режима для входного бита 00	ИСТИНА	Режим чтения зарегистрированного времени изменения входа: однократно (время первого изменения)

Карта входов-выходов

В таблице ниже перечислены используемые параметры карты соответствия входов и выходов (I/O map).

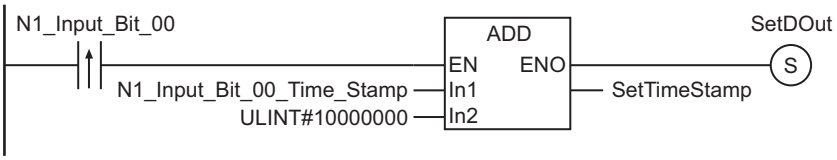
Расположение	Порт	Описание	Чтение/запись	Тип данных	Переменная	Тип переменной
Node1	Time Stamp of Synchronous Output	Содержит метку времени для синхронизации обновления синхронных выходов подключенного модуля NX. (ед.: нс).	Чт.	ULINT	E001_Time_Stamp_of_Synchronous_Output	Глобальная переменная
Unit1	Input Bit 00	Входной бит 00	Чт.	BOOL	N1_Input_Bit_00	Глобальная переменная
Unit1	Input Bit 00 Time Stamp	Время изменения входа для входного бита 00	Чт.	ULINT	N1_Input_Bit_00_Time_Stamp	Глобальная переменная
Unit2	Output Bit 00 Time Stamp	Указанное время для выходного бита 00	Зап.	ULINT	N2_Output_Bit_00_Time_Stamp	Глобальная переменная
Unit2	Output Bit 00	Выходной бит 00	Зап.	BOOL	N2_Output_Bit_00	Глобальная переменная

Программа на языке LD

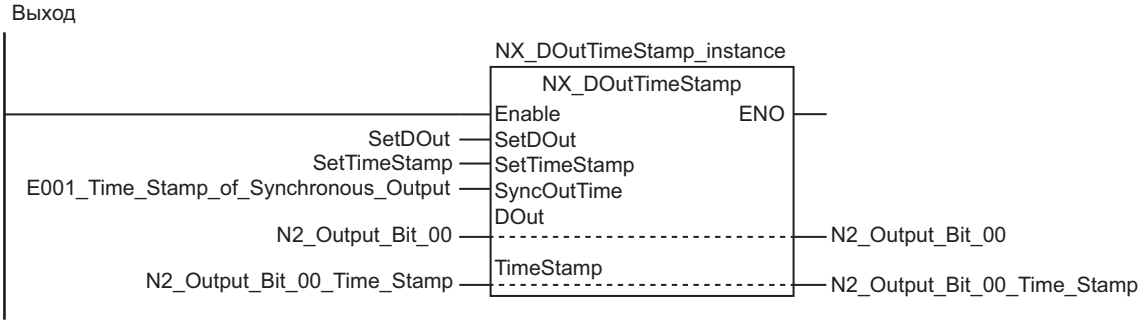
Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	SetTimeStamp	ULINT	0	Заданная метка времени
	SetDOut	BOOL	ЛОЖЬ	Выходное значение
	NX_DOutTimeStamp_instance	NX_DOutTimeStamp		

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	N1_Input_Bit_00	BOOL	☐	Входной бит 00
	N1_Input_Bit_00_Time_Stamp	ULINT	☐	Время изменения входа для входного бита 00
	E001_Time_Stamp_of_Synchronous_Output	ULINT	☐	Метка времени для синхронизации обновления синхронных выходов подключенного модуля NX
	N2_Output_Bit_00	BOOL	☐	Выходной бит 00
	N2_Output_Bit_00_Time_Stamp	ULINT	☐	Указанное время для выходного бита 00

Указание метки времени вывода.



Выход



Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	SetEN	BOOL	ЛОЖЬ	Условие выполнения
	SetTimeStamp	ULINT	0	Заданная метка времени
	SetDOut	BOOL	ЛОЖЬ	Выходное значение
	R_TRIG_instance	R_TRIG		
	NX_DOutTimeStamp_instance	NX_DOutTimeStamp		

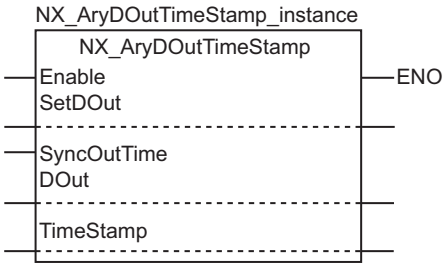
Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	N1_Input_Bit_00	BOOL	☐	Входной бит 00
	N1_Input_Bit_00_Time_Stamp	ULINT	☐	Время изменения входа для входного бита 00

Внешние переменные	Переменная	Тип данных	Константа	Комментарий
	E001_Time_Stamp_of_Synchronous_Output	ULINT	⌐¬	Метка времени для синхронизации обновления синхронных выходов подключенного модуля NX
	N2_Output_Bit_00	BOOL	⌐¬	Выходной бит 00
	N2_Output_Bit_00_Time_Stamp	ULINT	⌐¬	Указанное время для выходного бита 00

```
// Вход запуска выполнения
R_TRIG_instance( N1_Input_Bit_00, SetEN);
// Указание метки времени вывода.
IF ( SetEN = TRUE ) THEN
SetDOut := TRUE;
SetTimeStamp := N1_Input_Bit_00_Time_Stamp + ULINT#10000000;
END_IF;
// Вывод
NX_DOutTimeStamp_instance(
Enable := TRUE,
SetDOut := SetDOut,
SetTimeStamp := SetTimeStamp,
SyncOutTime := E001_Time_Stamp_of_Synchronous_Output,
DOut := N2_Output_Bit_00,
TimeStamp := N2_Output_Bit_00_Time_Stamp);
```

NX_AryDOutTimeStamp

Команда NX_AryDOutTimeStamp выдает последовательность импульсов с выхода модуля дискретных выходов, поддерживающего обновление с указанием метки времени.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
NX_AryDOutTimeStamp	Запись в дискретный выход из массива с указанием метки времени	FB		NX_AryDOutTimeStamp_instance(Enable, SetDOut, SyncOutTime, DOut, TimeStamp);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.06 или более поздней и Sysmac Studio версии 1.07 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Enable	Активация	Вход	ИСТИНА: выход изменяется в соответствии с настройкой *SetDOut*. ЛОЖЬ: когда значение этой переменной меняется на ЛОЖЬ, выходное значение также меняется на ЛОЖЬ.	Зависит от типа данных.	---	ЛОЖЬ
SyncOutTime	Метка времени синхронного вывода		Переменная устройства *Time Stamp of Synchronous Output (Метка времени синхронного вывода)* интерфейсного модуля EtherCAT или модуля NX в стойке модуля ЦПУ		нс	*1

	Значение	Вход/выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
SetDOut	Выходные импульсы	Вход-выход	Выходные импульсы	---	---	---
DOut	Выходной бит модуля дискретных выходов		Переменная устройства *Output Bit* ** модуля дискретных выходов, поддерживающего обновление с указанием метки времени	Зависит от типа данных.		
TimeStamp	Метка времени		Переменная устройства *Output Bit* ** *Time Stamp* модуля дискретных выходов, поддерживающего обновление с указанием метки времени		нс	

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	ОК																			
SyncOutTime									ОК											
SetDOut	Подробные сведения о структуре _sOUTPUT_REF см. в разделе *Определение выходных импульсов* на стр. 2-1645.																			
DOut	ОК																			
TimeStamp									ОК											

Функция

Когда значение переменной *Enable* = ИСТИНА, команда NX_AryDOutTimeStamp в заданное время выдает последовательность импульсов, заданную параметром *SetDOut* (выходные импульсы), с выхода модуля дискретных выходов, поддерживающего обновление с указанием метки времени.

Если вход *Enable* переходит в состояние ЛОЖЬ, команда NX_AryDOutTimeStamp выводит значение ЛОЖЬ в модуль дискретных выходов, поддерживающий обновление с указанием метки времени.
Разница между фактическим временем вывода и заданным временем не превышает ±1 мкс.

Значение *SyncOutTime* (метка времени синхронного вывода) определяется на основе информации часов в интерфейсном модуле EtherCAT или модуле NX, подключенном к шине NX модуля ЦПУ, к которому подключен модуль дискретных выходов, поддерживающий обновление с указанием метки времени.

Следует указывать переменную устройства *Time Stamp of Syncronous Output (Метка времени синхронного вывода)* интерфейсного модуля EtherCAT или модуля NX, подключенного к шине NX модуля ЦПУ, к которому подключен модуль дискретных выходов.
Однако в случае интерфейсного модуля EtherCAT к записям ввода-вывода необходимо добавлять 0x200A:02 (метка времени синхронного вывода).

В качестве параметра *DOut* (выходной бит модуля дискретных выходов) следует указывать переменную устройства *Output Bit* **, которая назначена выходному биту модуля дискретных выходов, поддерживающего обновление с указанием метки времени.

В качестве параметра *TimeStamp* следует указывать переменную устройства *Output Bit ** Time Stamp*, которая назначена метке времени выходного бита модуля дискретных выходов, поддерживающего обновление с указанием метки времени.

Указание времени вывода

Для указания времени вывода следует соблюдать приведенный ниже порядок действий.

1 Получите значение переменной устройства, которая назначена информации часов (служит в качестве опорного времени для бита модуля).

2 Вычислите разницу между полученной информацией часов и временем перевода выходного бита в состояние ВКЛ в наносекундах и добавьте ее к значению переменной устройства из шага 1.

3 Передайте результат сложения в переменную SetDOut.OnTime[] команды NX_AryDOutTimeStamp.

4 Таким же образом, как на шаге 2, вычислите разницу между полученной информацией часов и временем перевода выходного бита в состояние ВЫКЛ в наносекундах и добавьте ее к значению переменной устройства из шага 1.

5 Передайте результат сложения в переменную SetDOut.OffTime[] команды NX_AryDOutTimeStamp.

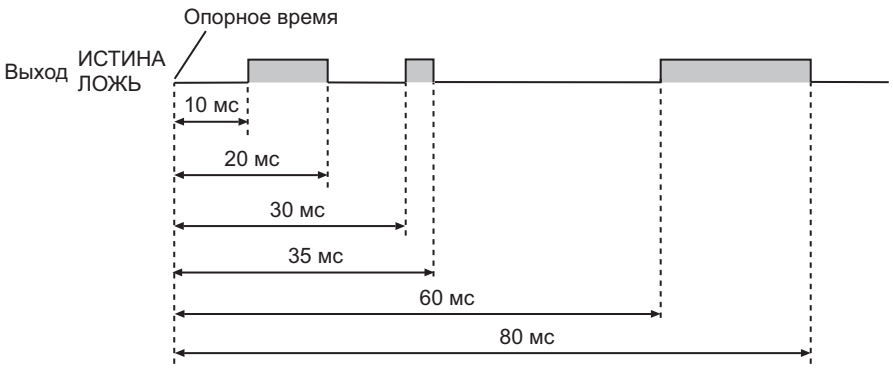
Определение выходных импульсов

Для определения параметров выходных импульсов служит переменная *SetDOut*, имеющая структурный тип данных _sOUTPUT_REF. Описание приведено в таблице ниже.

Имя	Значение	Описание	Тип данных	Диапазон допустимых значений	Единица	По умолчанию
SetDOut	Выходные импульсы	Выходные импульсы	_sOUTPUT_REF	---	---	---
EnableOut	Разрешение вывода	Флаг разрешения вывода ИСТИНА: активировать параметры *OnTime* и *OffTime*. ЛОЖЬ: деактивировать параметры *OnTime* и *OffTime*.	BOOL	Зависит от типа данных.	---	ЛОЖЬ
Массив OnTime[]	Значения времени включения	Моменты времени, в которые выходной бит переводится в состояние ВКЛ	ARRAY[0..15] OF ULINT		нс	0 для всех элементов
Массив OffTime[]	Значения времени выключения	Моменты времени, в которые выходной бит переводится в состояние ВЫКЛ	ARRAY[0..15] OF ULINT			

Массивы OnTime[] (значения времени включения) и OffTime[] (значения времени выключения) содержат по 16 элементов каждый.
Каждый элемент соответствующего массива представляет время включения и время выключения для одного импульса.
Соответственно, с помощью элементов двух массивов можно задать до 16 импульсов.
Если некоторая пара элементов с одинаковым номером в двух массивах содержит значение 0, значения всех последующих элементов недействительны.
На рисунке ниже показан пример работы выхода при указанных в таблице ниже значениях элементов массивов OnTime[] и OffTime[]. Указанные в таблице значения времени — это количество миллисекунд относительно опорного времени.

Имя	Номера элементов				
	0	1	2	3	4
OnTime[]	Спустя 10 мс	Спустя 30 мс	Спустя 60 мс	0	Спустя 90 мс
OffTime[]	Спустя 20 мс	Спустя 35 мс	Спустя 80 мс	0	Спустя 100 мс



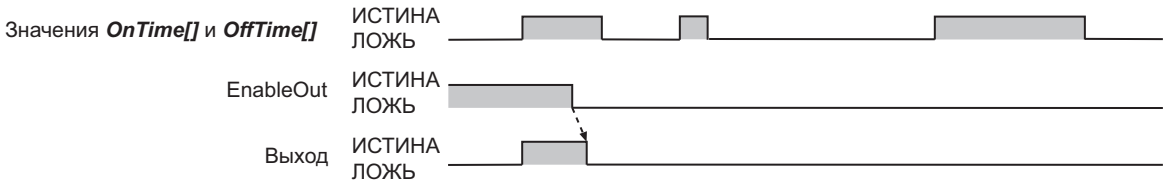
Значения элементов массивов OnTime[] и OffTime[] не обязательно указывать в хронологическом порядке. Например, при следующих значениях элементов массивов OnTime[] и OffTime[] выход будет работать точно так же, как и при значениях, приведенных выше.

Имя	Номера элементов				
	0	1	2	3	4
OnTime[]	Спустя 30 мс	Спустя 60 мс	Спустя 10 мс	0	Спустя 90 мс
OffTime[]	Спустя 35 мс	Спустя 80 мс	Спустя 20 мс	0	Спустя 100 мс

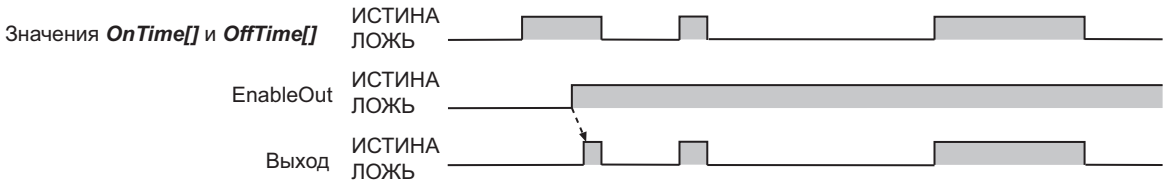
● *EnableOut* (разрешение вывода)

Флаг *EnableOut* (разрешение вывода) приводит в действие (активирует) значения, заданные в массивах OnTime[] и OffTime[].
Если *EnableOut* = ЛОЖЬ, на выход подается значение ЛОЖЬ независимо от значений элементов в OnTime[] и OffTime[].

Значение *EnableOut* можно изменять во время выполнения команды.
Когда вход *EnableOut* переходит в состояние ЛОЖЬ, выходное значение также меняется на ЛОЖЬ.

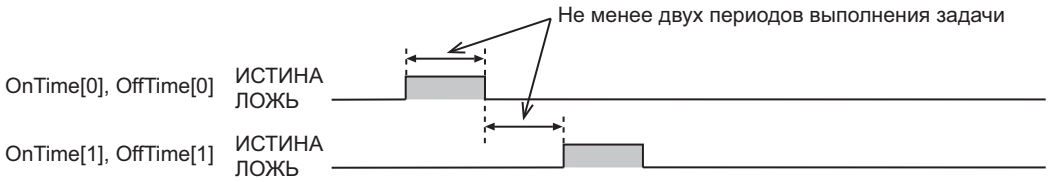


Когда вход *EnableOut* переходит в состояние ИСТИНА, вступают в силу настройки в OnTime[] и OffTime[].



● Минимальная ширина выходного импульса

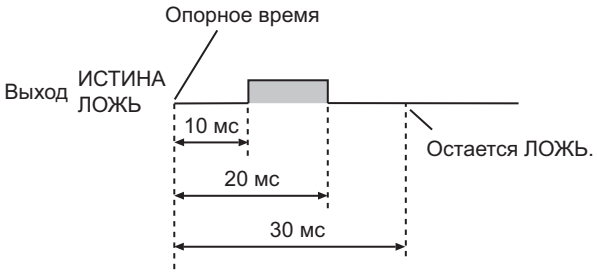
Чтобы погрешность времени вывода импульсов не превышала 1 мкс, интервал между любыми двумя соседними значениями OnTime[] и OffTime[] должен не менее чем в два раза превосходить длительность цикла выполнения задачи.
При интервале менее двух периодов выполнения задачи импульсы могут выдаваться не так, как задано, или могут выдаваться с задержкой на один цикл задачи относительно заданного времени включения или выключения.



● Когда элементы массивов OnTime[] и OffTime[] с одинаковым номером содержат одинаковое значение

Когда некоторая пара элементов с одинаковым номером в массивах OnTime[] и OffTime[] содержит одинаковое значение, на выход подается значение ЛОЖЬ. На рисунке ниже показана работа выхода при указанных в таблице ниже значениях элементов двух массивов.

Имя	Номера элементов		
	0	1	2
OnTime[]	Спустя 10 мс	Спустя 30 мс	0
OffTime[]	Спустя 20 мс	Спустя 30 мс	0



● Когда значение элемента в массиве OnTime[] больше значения в массиве OffTime[]

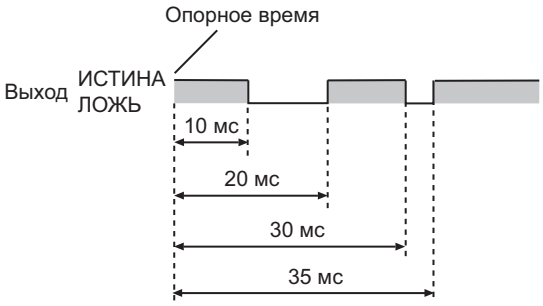
Если значение некоторого элемента в массиве OnTime[] больше значения соответствующего элемента массива OffTime[], выход перейдет в состояние ЛОЖЬ, а затем снова вернется в состояние ИСТИНА.

Если наименьшее из значений элементов массива OnTime[] больше, чем наименьшее из значений элементов массива OffTime[], выход перейдет в состояние ИСТИНА сразу после выполнения этой команды.

Также, если наибольшее из значений элементов массива OnTime[] больше, чем наибольшее из значений элементов массива OffTime[], выход останется в состоянии ИСТИНА после выполнения этой команды.

На рисунке ниже показана работа выхода при указанных в таблице ниже значениях элементов двух массивов.

Имя	Номера элементов		
	0	1	2
OnTime[]	Спустя 20 мс	Спустя 35 мс	0
OffTime[]	Спустя 10 мс	Спустя 30 мс	0

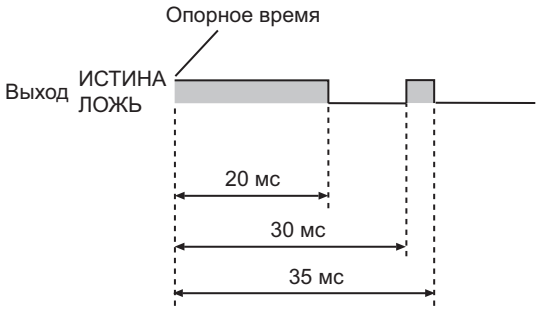


● Когда значение одного элемента в массиве OnTime[] или в массиве OffTime[] равно 0

Если в массиве OnTime[] или OffTime[] имеется элемент со значением 0, выход перейдет в состояние ИСТИНА или ЛОЖЬ сразу после выполнения этой команды.

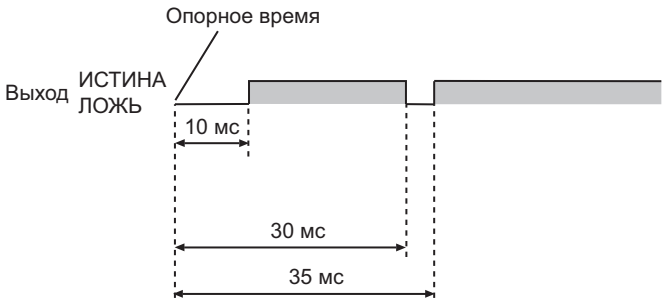
Если какой-либо элемент массива OnTime[] содержит значение 0, выход перейдет в состояние ИСТИНА сразу после выполнения этой команды. На рисунке ниже показана работа выхода при указанных в таблице ниже значениях элементов двух массивов.

Имя	Номера элементов		
	0	1	2
OnTime[]	0	Спустя 30 мс	0
OffTime[]	Спустя 20 мс	Спустя 35 мс	0



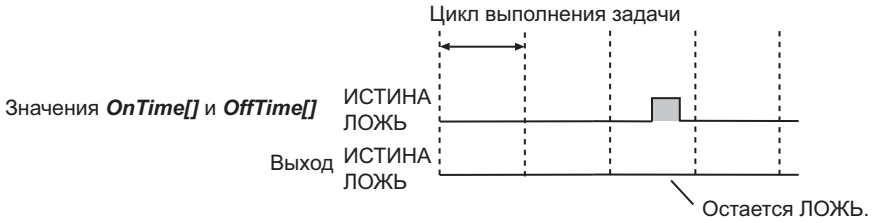
Если какой-либо элемент массива OffTime[] содержит значение 0, выход перейдет в состояние ЛОЖЬ сразу после выполнения этой команды. На рисунке ниже показана работа выхода при указанных в таблице ниже значениях элементов двух массивов.

Имя	Номера элементов		
	0	1	2
OnTime[]	Спустя 10 мс	Спустя 35 мс	0
OffTime[]	0	Спустя 30 мс	0



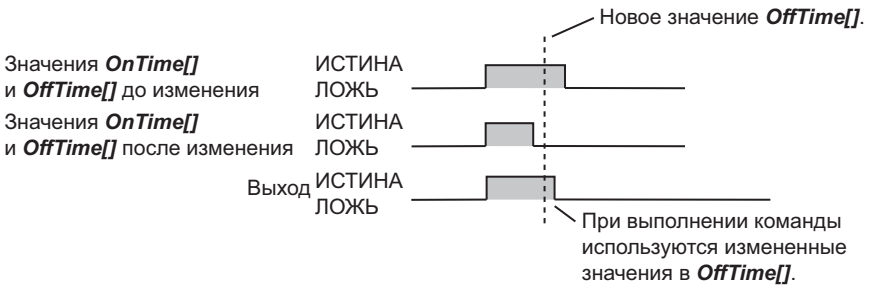
● Когда для выхода настроен переход в состояние ИСТИНА и возврат в состояние ЛОЖЬ в пределах одного периода задачи

Если значения времени включения и выключения для выхода настроены таким образом, что переход в состояние ИСТИНА и возврат в состояние ЛОЖЬ происходят в одном цикле выполнения задачи, значение выхода вообще не изменяется.



● Изменение значений в массиве OnTime[] или OffTime[] при значении ИСТИНА на входе Enabled

Значения элементов массивов OnTime[] и OffTime[] можно изменять в том числе и тогда, когда вход Enabled находится в состоянии ИСТИНА (т. е. когда команда активирована).
Новые значения вступят в силу при следующем выполнении команды.



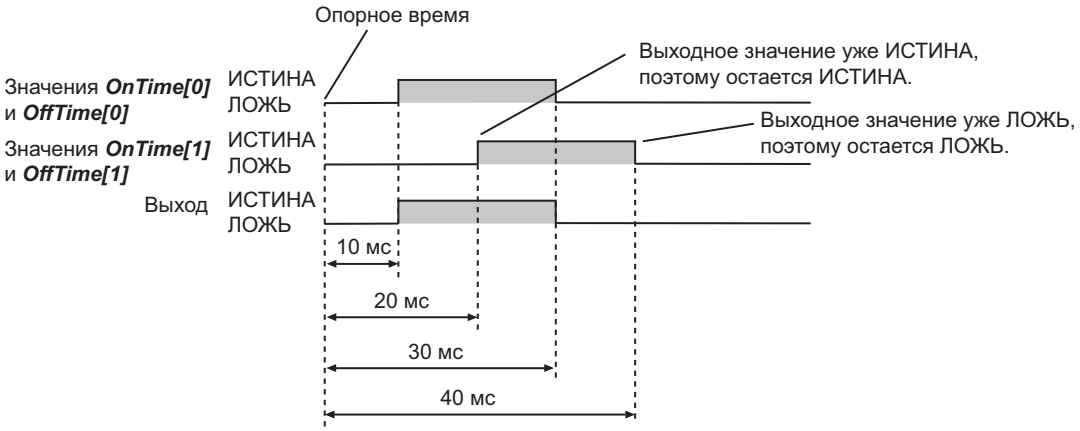
● Перекрывающиеся значения времени переключения выхода в состояние ИСТИНА

Если заданные значения времени перехода выхода в состояние ИСТИНА перекрываются, ошибки не происходит и выход остается в состоянии ИСТИНА.

Та же логика применима и к случаю, когда перекрываются значения времени перехода выхода в состояние ЛОЖЬ.

На рисунке ниже показана работа выхода при указанных в таблице ниже значениях элементов двух массивов.

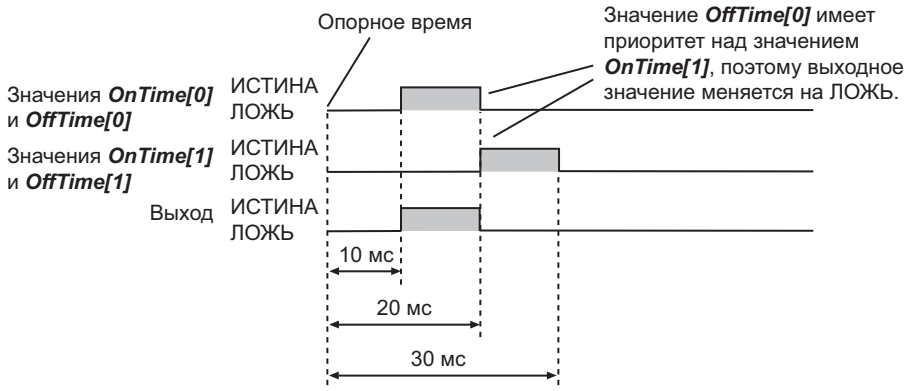
Имя	Номера элементов		
	0	1	2
OnTime[]	Спустя 10 мс	Спустя 20 мс	0
OffTime[]	Спустя 30 мс	Спустя 40 мс	0



● Одинаковые значения времени переключения выхода для состояний ИСТИНА и ЛОЖЬ

Если для выхода задано одно и то же время для переключения в состояние ИСТИНА и для переключения в состояние ЛОЖЬ, ошибки не произойдет и более приоритетным в этом случае будет элемент массива OnTime[] или OffTime[] с меньшим номером элемента. На рисунке ниже показана работа выхода при указанных в таблице ниже значениях элементов двух массивов.

Имя	Номера элементов		
	0	1	2
OnTime[]	Спустя 10 мс	Спустя 20 мс	0
OffTime[]	Спустя 20 мс	Спустя 30 мс	0



Дополнительная информация

Данная команда используется совместно с командой MC_DigitalCamSwitch.
Подробное описание команды MC_DigitalCamSwitch см. в документе *Серия NJ/NX — Команды программирования для управления движением. Справочное руководство (Cat. No. W508)*.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Эту команду можно выполнить только для модуля дискретных выходов, который поддерживает обновление с использованием метки времени. Но даже если в системе нет ни одного подключенного модуля дискретных выходов, поддерживающего обновление с использованием метки времени, и будет выполнена эта команда, к возникновению ошибки это не приведет.
- Если произойдет ошибка связи по сети EtherCAT или будет превышен период выполнения задачи, вывод может быть не выполнен в указанное время. В этом случае значение выводится в следующем цикле выполнения задачи или еще позже.
- Если какая-либо переменная устройства, используемая для этой команды, изменяется или читается любой другой командой или программой, необходимо предусмотреть эксклюзивное управление.
- Параметр *SyncOutTime* следует указывать с помощью переменной устройства *Time Stamp of Syncronous Output (Метка времени синхронного вывода)* интерфейсного модуля EtherCAT или модуля NX, подключенного к шине NX модуля ЦПУ, к которому подключен модуль дискретных выходов, поддерживающий обновление с указанием метки времени.
  Но даже если будет указана другая переменная, ошибки не произойдет.
- В качестве параметров *DOut* и *TimeStamp* следует указывать соответствующие переменные устройства модуля дискретных выходов, поддерживающего обновление с указанием метки времени. Но даже если будет указана другая переменная, ошибки не произойдет.
- В качестве параметров *DOut* и *TimeStamp* следует указывать соответствующие переменные устройства для того же канала того же модуля. Но даже если будет указана другая переменная, ошибки не произойдет.
- Если для переменной *TimeStamp* задано время в прошлом, значение *TimeStamp* становится равно 0.
  В этом случае состояние выхода модуля дискретных выходов, поддерживающего обновление с указанием метки времени, обновляется немедленно.
  Дополнительные сведения см. в документе *Серия NX, модули дискретных входов-выходов — Руководство пользователя (Cat. No. W521)*.

Пример программы

Пример программы см. в описании команды MC_DigitalCamSwitch в документе *Серия NJ/NX — Команды программирования для управления движением. Справочное руководство (Cat. No. W508)*.

Прочие команды

Команда	Имя	Стр.
ReadNbit_**	Группа чтения N битов	стр. 2-1654
WriteNbit_**	Группа записи N битов	стр. 2-1656
ChkRange	Проверка на принадлежность заданному диапазону	стр. 2-1658
GetMyTaskStatus	Чтение состояния текущей задачи	стр. 2-1661
GetMyTaskInterval	Чтение периода текущей задачи	стр. 2-1664
Task_IsActive	Определение состояния задачи	стр. 2-1667
Lock и Unlock	Блокировка задач/Разблокировка задач	стр. 2-1669
ActEventTask	Инициация событийной задачи	стр. 2-1675
Get**Clk	Группа получения тактовых импульсов	стр. 2-1682
Get**Cnt	Группа получения значения автономного прямого счетчика	стр. 2-1684

ReadNbit_**

Команды ReadNbit_** считывают состояния ноля или большего числа битов в битовой строке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ReadNbit_**	Группа чтения N битов	FUN	(@)ReadNbit_** EN ENO In Out Pos Size "**" — тип данных, относящийся к битовым строкам.	Out:=ReadNbit_**(In, Pos, Size); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник для чтения	Вход	Битовая строка для чтения	Зависит от типа данных.	---	0
Pos	Позиция чтения		Позиция бита для чтения	От 0 до количества битов в *In* -1		
Size	Размер чтения		Количество битов для чтения	От 0 до количества битов в *In*		1
Out	Результат чтения	Выход	Результат чтения	Зависит от типа данных.	---	---

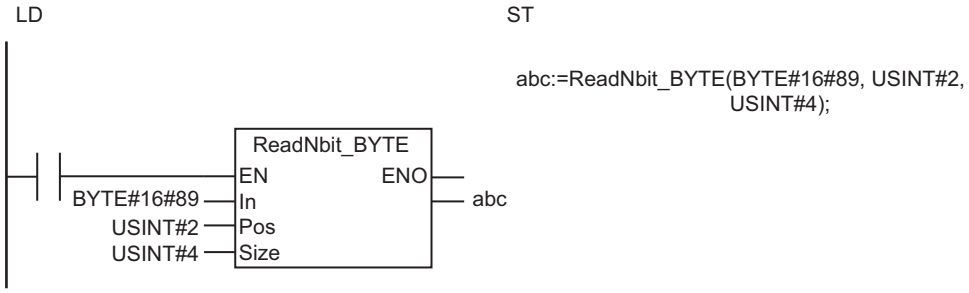
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		ОК	ОК	ОК	ОК															
Pos						ОК														
Size						ОК														
Out	Тип данных должен быть таким же, как у *In*.																			

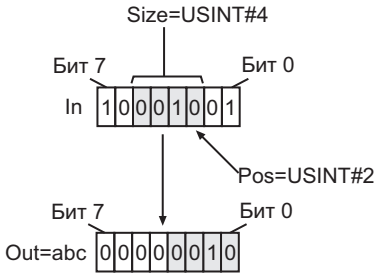
Функция

Команда ReadNbit_** считывает значения старших *Size* битов в позиции *Pos* (позиция чтения) в *In* (источник для чтения), а затем присваивает значения переменной *Out* (результат чтения).

Имя команды определяется типом данных *In* и *Out*. Например, если *In* и *Out* относятся к типу данных WORD, команда будет иметь имя ReadNbit_WORD.

Ниже приведен пример работы команды ReadNbit_BYTE для случая, когда *In* = BYTE#16#89, *Pos* = USINT#2, а *Size* = USINT#4.





Дополнительная информация

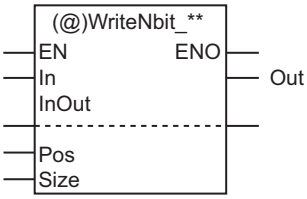
Для записи значений в ноль или большее число битов используйте команду *WriteNbit_*** на стр. 2-1656.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных *In* и *Out* должны быть одинаковыми.
- Если значение *Size* равно 0, то значение *Out* также равно 16#0.
- В указанных ниже случаях произойдет ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *Out* не изменится.
  a) Значение *Size* находится за пределами допустимого диапазона.
  b) Значение *Pos* находится за пределами допустимого диапазона.
  c) В переменной *In* нет столько битов после позиции *Pos*, сколько указано в *Size*.

WriteNbit_**

Команды WriteNbit_** производят запись в ноль или большее число битов в битовой строке.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
WriteNbit_**	Группа записи N битов	FUN	 "**" — тип данных, относящийся к битовым строкам.	WriteNbit_**(In, InOut, Pos, Size); "**" — тип данных, относящийся к битовым строкам.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Источник для чтения	Вход	Битовая строка, из которой считываются состояния битов для записи в *InOut*	Зависит от типа данных.	---	0
Pos	Позиция записи		Позиция бита, в которую производится запись	От 0 до количества битов в *InOut* -1		
Size	Размер записи		Количество битов для записи	От 0 до количества битов в *In*		1
InOut	Целевой объект для записи	Вход-выход	Результат записи	Зависит от типа данных.	---	---
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		ОК	ОК	ОК	ОК															
Pos						ОК														
Size						ОК														
InOut	Тип данных должен быть таким же, как у *In*.																			
Out	ОК																			

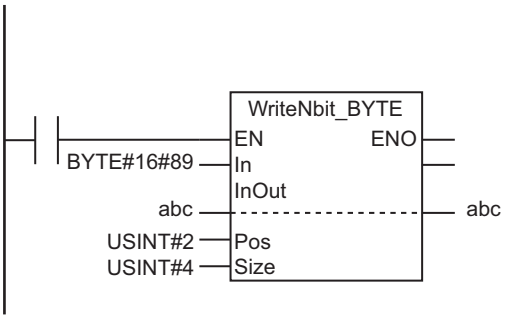
Функция

Команда WriteNbit_** сначала считывает значения *Size* младших битов из *In* (источник для чтения). Затем она записывает прочитанные значения в позицию *Pos* (позиция записи) в *InOut* (целевой объект для записи).

Имя команды определяется типом данных *In* и *Out*. Например, если *In* и *Out* относятся к типу данных WORD, команда будет иметь имя WriteNbit_WORD.

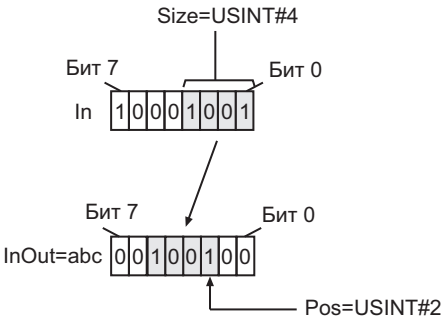
Ниже приведен пример работы команды WriteNbit_BYTE для случая, когда *In* = BYTE#16#89, *Pos* = USINT#2, а *Size* = USINT#4.

LD



ST

```
WriteNbit_BYTE(BYTE#16#89, abc,
               USINT#2, USINT#4);
```



Дополнительная информация

Для чтения значений ноля или большего числа битов используйте команду *ReadNbit_** на стр. 2-1654.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Типы данных *In* и *InOut* должны быть одинаковыми.
- Значение выхода *InOut* не изменяется, если *Size* = 0.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.
- В указанных ниже случаях происходит ошибка. Выход *ENO* будет содержать ЛОЖЬ, а содержимое *InOut* не изменится.
  a) Значение *Size* находится за пределами допустимого диапазона.
  b) Значение *Pos* находится за пределами допустимого диапазона.
  c) В битовой строке в *InOut* нет столько битов после позиции *Pos*, сколько указано в *Size*.

ChkRange

Команда ChkRange определяет, находится ли значение переменной в пределах заданного диапазона значений.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ChkRange	Проверка на принадлежность заданному диапазону	FUN	(@)ChkRange EN In Val Out	Out:=ChkRange(In, Val);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
In	Переменная для проверки	Вход	Проверяемая переменная	Зависит от типа данных.	---	*1
Val	Переменная задания диапазона		Переменная, в определении типа данных которой задан диапазон значений	Зависит от указанного диапазона.		
Out	Результат проверки	Выход	Результат проверки	Зависит от типа данных.	---	---

*1. Если входной параметр опущен, значение по умолчанию не применяется. В этом случае произойдет ошибка сборки.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In						ОК	ОК	ОК	ОК	ОК	ОК	ОК	ОК							
Val	Базовый тип данных, лежащий в основе указания диапазона, должен быть таким же, как у *In*.																			
Out	ОК																			

Функция

Команда ChkRange определяет, находится ли значение *In* (переменная для проверки) в пределах допустимого диапазона значений, заданного переменной *Val* (переменная задания диапазона).

Если значение находится в допустимом диапазоне, выход *Out* (результат проверки) переходит в состояние ИСТИНА. Если значение выходит за допустимый диапазон, переменная результата проверки принимает значение ЛОЖЬ.

Дополнительная информация

Диапазон допустимых значений можно указывать в определении типа для целочисленных переменных (USINT, UINT, UDINT, ULINT, SINT, INT, DINT и LINT).

Меры предосторожности для обеспечения надлежащей эксплуатации

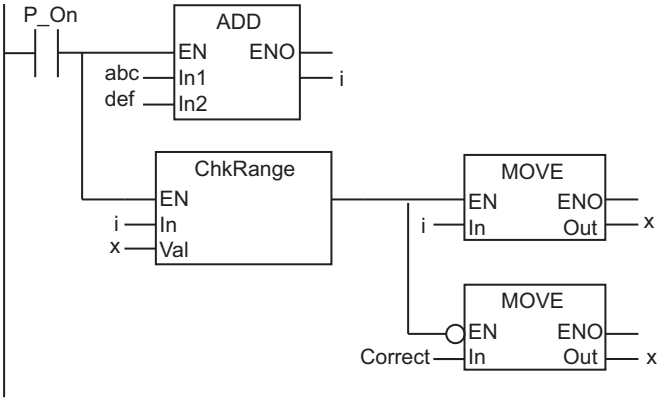
- Если *Val* не является переменной задания диапазона, значение *Out* меняется на ИСТИНА.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей команде данной логической цепи происходит ошибка.

Пример программы

Ниже представлены примеры, в которых проверяется, не выходит ли результат операции сложения *i* за диапазон допустимых значений (10...99), который задан в определении типа переменной *x*.
Если результат находится в пределах допустимого диапазона, значение *i* присваивается переменной *x*. Если результат выходит за допустимый диапазон, переменной *x* присваивается значение переменной *Correct*.

Программа на языке LD

Переменная	Тип данных	Начальное значение
i	INT	0
abc	INT	0
def	INT	0
x	INT(10..99)	10
Correct	INT	0



Программа на языке ST

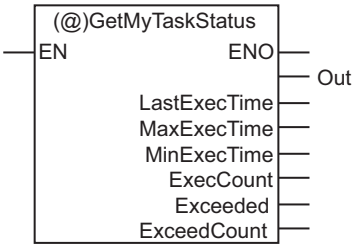
Переменная	Тип данных	Начальное значение
i	INT	0
abc	INT	0
def	INT	0
Chk	BOOL	ЛОЖЬ

Переменная	Тип данных	Начальное значение
x	INT(10..99)	10
Correct	INT	0

```
i := abc+def;
Chk:=ChkRange(i, x);   // Проверка на принадлежность заданному диапазону.

IF (Chk=TRUE) THEN
    x := i;            // i присваивается x, если значение i в диапазоне.
ELSE
    x := Correct;      // Correct присваивается x, если значение i не в диапазоне.
END_IF;
```

GetMyTaskStatus

Команда GetMyTaskStatus считывает состояние текущей задачи.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetMyTaskStatus	Чтение состояния текущей задачи	FUN		GetMyTaskStatus(LastExecTime, MaxExecTime, MinExecTime, ExecCount, Exceeded, ExceedCount);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---
LastExecTime	Время последнего выполнения задачи		Время, в течение которого текущая задача выполнялась в последний раз.	Зависит от типа данных.*1	нс	
MaxExecTime	Максимальное время выполнения задачи		Максимальное время выполнения текущей задачи			
MinExecTime	Минимальное время выполнения задачи		Минимальное время выполнения текущей задачи			
ExecCount	Количество выполнений задачи		Количество выполнений текущей задачи	Зависит от типа данных.	---	
Exceeded	Флаг превышения периода задачи		ИСТИНА: последнее выполнение текущей задачи не было завершено за один цикл выполнения задачи. ЛОЖЬ: последнее выполнение текущей задачи было завершено за один цикл выполнения задачи.			
ExceedCount	Количество превышений периода задачи		Количество раз, когда текущая задача не успела завершиться за один цикл выполнения задачи.			

*1. Отрицательные числа исключены.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out						ОК														
LastExecTime																ОК				
MaxExecTime																ОК				
MinExecTime																ОК				
ExecCount								ОК												
Exceeded	ОК																			
ExceedCount								ОК												

Функция

Команда GetMyTaskStatus считывает состояние текущей задачи.
Считываемое состояние задачи включает в себя следующую информацию: время последнего выполнения задачи *LastExecTime*, максимальное время выполнения задачи *MaxExecTime*, минимальное время выполнения задачи *MinExecTime*, количество выполнений задачи *ExecCount*, флаг превышения периода задачи *Exceeded* и количество превышений периода задачи *ExceedCount*.

Дополнительная информация

Ниже указано, когда происходит сброс значений *MaxExecTime*, *MinExecTime*, *ExecCount* и *ExceedCount*.

- В начале работы.
- При выполнении операции сброса в окне Task Execution Time Monitor (Контроль времени выполнения задач) в Sysmac Studio.

Меры предосторожности для обеспечения надлежащей эксплуатации

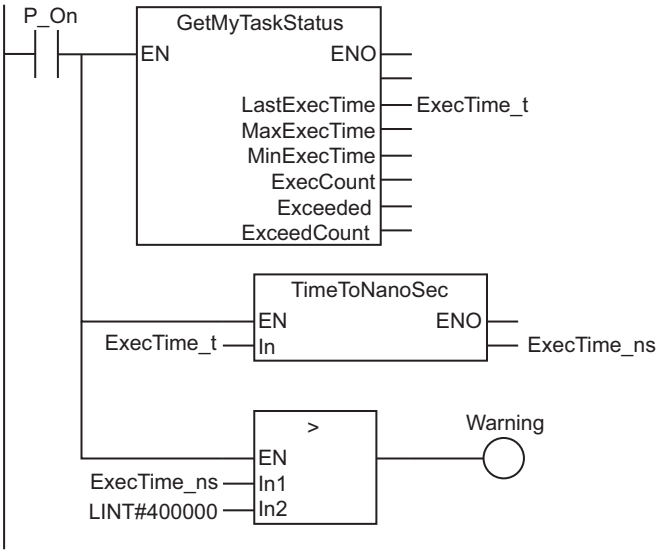
- Когда значение *ExecCount* или *ExceedCount* превышает максимально возможное значение типа данных UDINT (4 294 967 295), оно возвращается к 0.
- При использовании команды в программе на языке ST возвращаемое значение *Out* не используется.

Пример программы

Ниже показан пример применения команды GetMyTaskStatus для чтения состояния текущей задачи.
Если время предыдущего выполнения задачи превышает 400 мкс (400 000 нс), значение переменной *Warning* меняется на ИСТИНА.

Программа на языке LD

Переменная	Тип данных	Начальное значение	Комментарий
ExecTime_t	TIME	T#0s	Время предыдущего выполнения задачи (значение типа TIME)
ExecTime_ns	LINT	0	Время предыдущего выполнения задачи (количество наносекунд, значение типа LINT)
Warning	BOOL	ЛОЖЬ	Предупреждение



Программа на языке ST

Переменная	Тип данных	Начальное значение	Комментарий
ExecTime_t	TIME	T#0s	Время предыдущего выполнения задачи (значение типа TIME)
ExecTime_ns	LINT	0	Время предыдущего выполнения задачи (количество наносекунд, значение типа LINT)
Warning	BOOL	ЛОЖЬ	Предупреждение

```
GetMyTaskStatus(LastExecTime=>ExecTime_t);  // Получение предыдущего периода задачи
ExecTime_ns:=TimeToNanoSec(ExecTime_t);  // Преобразование значения предыдущего пер
иода задачи из типа данных TIME в наносекунды.
IF (ExecTime_ns>DINT#400000) THEN  // Если предыдущий период задачи превышает 400 0
00 нс...
    Warning:=TRUE;  // Присвоить ИСТИНА переменной Warning.
ELSE
    Warning:=FALSE;
END_IF;
```

GetMyTaskInterval

Команда GetMyTaskInterval служит для чтения периода выполнения текущей задачи.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
GetMyTaskInterval	Чтение периода текущей задачи	FUN	(@)GetMyTaskInterval EN ENO Out	Out:=GetMyTaskInterval();

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.08 или более поздней и Sysmac Studio версии 1.09 или выше.

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Период выполнения задачи	Выход	Период выполнения текущей задачи	Зависит от типа данных.*1	мс	---

*1. Отрицательные числа исключены.

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out																OK				

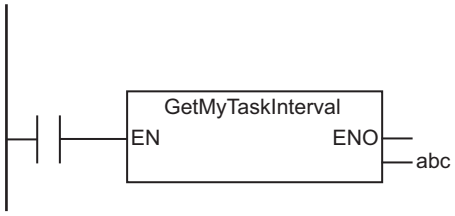
Функция

Команда GetMyTaskInterval производит чтение периода выполнения текущей задачи и сохраняет его в переменную *Out* (период выполнения текущей задачи), если задача, в которой выполняется эта команда, является главной циклической задачей или циклической задачей.

Если эта команда будет выполнена в событийной задаче, значение переменной *Out* будет равно T#0 с.

Пример программы представлен на рисунке ниже. Если период выполнения текущей задачи равен 1 мс, то значение переменной *abc* будет равно T#1 мс.

LD



ST

```
abc:=GetMyTaskInterval();
```


Пример программы

В данном примере программа считывает значение периода выполнения текущей задачи, когда эта программа выполняется в первый раз после начала работы контролера. Прочитанное значение периода выполнения задачи преобразуется из типа данных TIME в значение типа LREAL, содержащее количество миллисекунд.
Такую программу, например, можно использовать для вычисления целевого положения оси в каждом цикле выполнения задачи.

Для преобразования значения типа TIME в значение типа LREAL в миллисекундах используется приведенная ниже процедура.

1 С помощью команды GetMyTaskInterval считывается период выполнения задачи в виде значения типа TIME.

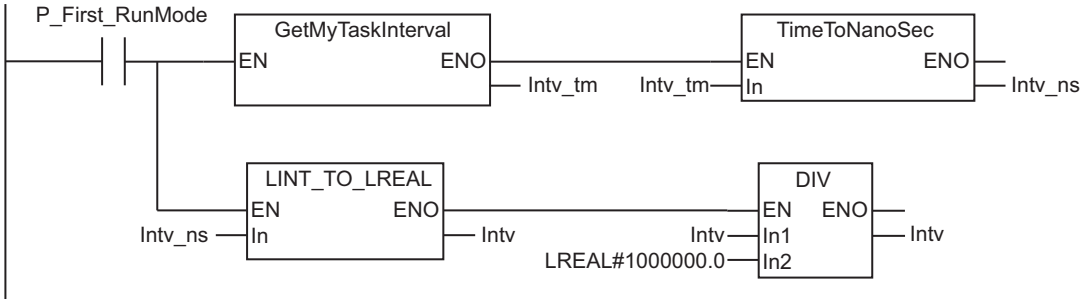
2 Используется команда TimeToNanoSec для преобразования значения типа TIME в значение типа LINT в наносекундах.

3 Используется команда LINT_TO_LREAL для преобразования значения типа LINT в наносекундах в значение типа LREAL в наносекундах.

4 С помощью команды DIV полученный на шаге 3 результат делится на 1 000 000 для перевода в миллисекунды.

Программа на языке LD

Переменная	Тип данных	По умолчанию	Комментарий
Intv_tm	TIME	T#0s	Период выполнения задачи в виде значения типа TIME
Intv_ns	LINT	0	Период выполнения задачи в виде значения типа LINT в наносекундах
Intv	LREAL	0	Период выполнения задачи в виде значения типа LREAL в миллисекундах



Программа на языке ST

Переменная	Тип данных	По умолчанию	Комментарий
Intv	LREAL	0	Период выполнения задачи в виде значения типа LREAL в миллисекундах

```
IF P_First_RunMode = TRUE THEN
   Intv := LINT_TO_LREAL(TimeToNanoSec(GetMyTaskInterval()))/1000000;
END_IF;
```

Task_IsActive

Команда Task_IsActive позволяет определить, выполняется ли в данный момент указанная задача.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Task_IsActive	Определение состояния задачи	FUN	(@)Task_IsActive EN TaskName Out	Out:=Task_IsActive(TaskName);

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
TaskName	Имя задачи	Вход	Имя задачи	Макс. 64 байт (63 однобайтовых буквенно-цифровых символа + последний символ NULL)	---	''
Out	Решение	Выход	ИСТИНА: задача выполняется или находится в режиме ожидания. ЛОЖЬ: задача не активна	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TaskName																				OK
Out	OK																			

Функция

Команда Task_IsActive позволяет определить, выполняется ли или находится ли в режиме ожидания в данный момент задача, указанная параметром *TaskName*.
"В режиме ожидания" означает, что после запуска этой задачи была запущена более приоритетная задача, поэтому выполнение данной задачи было прервано.
Если задача выполняется или находится в режиме ожидания, значение переменной *Out* (решение) равно ИСТИНА. Если задача не выполняется, значение переменной *Out* равно ЛОЖЬ.

Меры предосторожности для обеспечения надлежащей эксплуатации

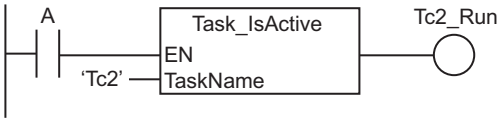
- В параметр *TaskName* нельзя передать переменную, содержащую текстовую строку. Необходимо указать непосредственно текстовую строку.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей логической цепи происходит ошибка.
- В указанном ниже случае произойдет ошибка. Значение *Out* при этом не изменится.
  a) Задачи с указанным именем *TaskName* не существует.

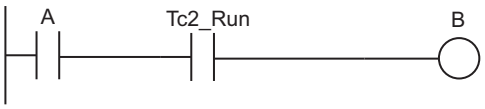
Пример программы

Рассмотрим пример, в котором данная команда определяет, активна ли циклическая задача Tc2, когда значение переменной *A* меняется на ИСТИНА. Если задача активна, то значение переменной *B* меняется на ИСТИНА.

Программа на языке LD

Переменная	Тип данных	Начальное значение	Комментарий
A	BOOL	ЛОЖЬ	
B	BOOL	ЛОЖЬ	
Tc2_Run	BOOL	ЛОЖЬ	Состояние выполнения задачи Tc2





Программа на языке ST

Переменная	Тип данных	Начальное значение	Комментарий
A	BOOL	ЛОЖЬ	
B	BOOL	ЛОЖЬ	
Tc2_Run	BOOL	ЛОЖЬ	Состояние выполнения задачи Tc2

```
IF (A=TRUE) THEN
    // Определение состояния задачи.
    Tc2_Run:=Task_IsActive('Tc2');
    // Присвоить ИСТИНА переменной B, если Tc2 выполняется.
    IF (Tc2_Run=TRUE) THEN
        B := TRUE;
    END_IF;

END_IF;
```

Lock и Unlock

Lock : Означает начало эксклюзивной блокировки между задачами. Выполнение любой другой задачи, включающей область блокировки с таким же номером блокировки, становится невозможно.

Unlock : Означает конец эксклюзивной блокировки между задачами.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Lock	Блокировка задач	FUN	(@)Lock EN ENO Index Out	Lock(Index);
Unlock	Разблокировка задач	FUN	(@)Unlock EN ENO Index Out	Unlock(Index);

Переменные

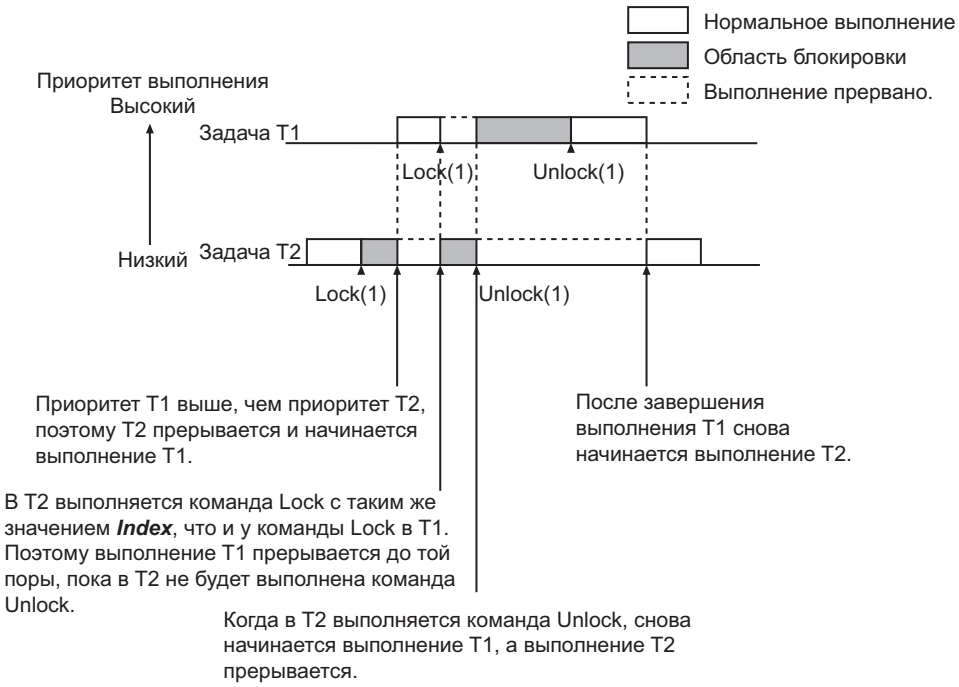
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Index	Номер блокировки	Вход	Номер блокировки	Зависит от типа данных.	---	0
Out	Возвращаемое значение	Выход	Всегда ИСТИНА	Только ИСТИНА	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Index						OK														
Out	OK																			

Функция

Команды Lock и Unlock создают области блокировки. Если в некоторой задаче в данный момент выполняется область блокировки, во всех остальных задачах области блокировки с тем же номером блокировки не выполняются.
Номер блокировки указывается параметром *Index*.

Пример программы представлен на рисунке ниже.
В задачах T1 и T2 имеется область блокировки со значением *Index*, равным 1. Если команда Lock сначала выполняется в задаче T2, область блокировки в задаче T1 не будет выполняться до тех пор, пока в задаче T2 не будет выполнена команда Unlock.



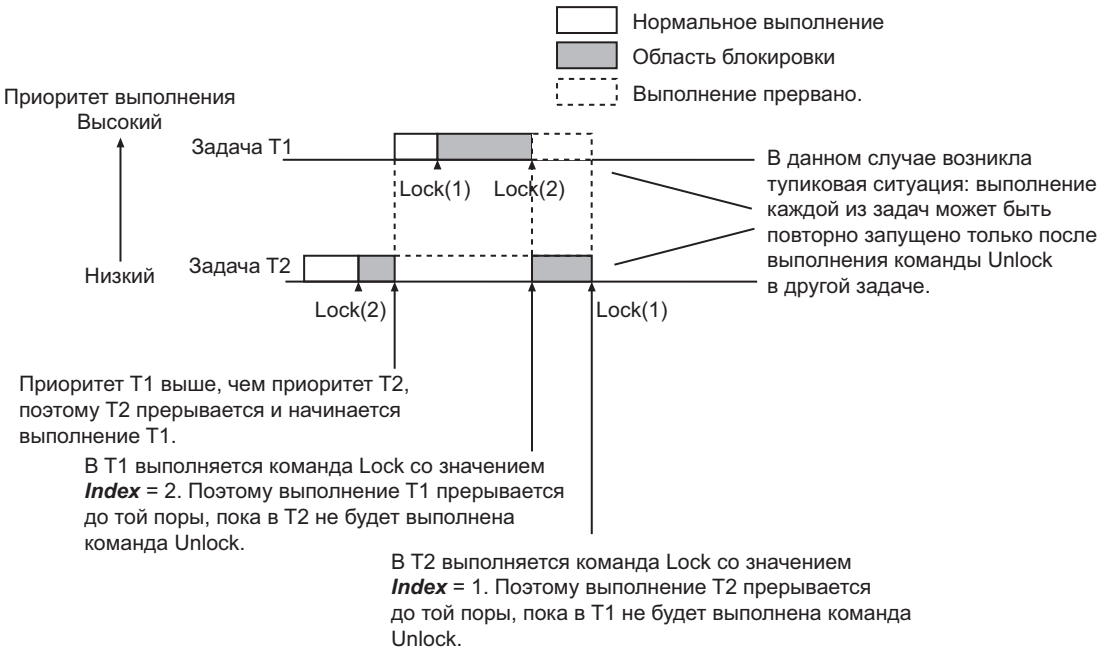
Области блокировки с разными значениями *Index* не влияют друг на друга.

Дополнительная информация

- Команды Lock и Unlock используются в случае, когда из нескольких разных задач необходимо производить чтение или запись одних и тех же данных.
  Они позволяют исключить одновременное обращение к одним и тем же данным из нескольких задач (чтобы задача не могла выполнить чтение или запись данных, когда эти данные читаются или записываются другой задачей).
- В одном программном компоненте можно разместить больше одной пары команд Lock и Unlock при условии, что у них различаются значения *Index*. Пары команд также могут вкладываться друг в друга.

Меры предосторожности для обеспечения надлежащей эксплуатации

- Не создавайте области блокировки длиннее, чем это необходимо. Если область блокировки будет слишком длинной, период выполнения задачи может возрасти.
- Всегда используйте команды Lock и Unlock в паре в пределах одной секции одного программного компонента.
- Всего может быть создано максимум 16 777 215 областей блокировки.
- Если команды Lock используются в нескольких задачах и неудачно расположены относительно друг друга, может возникнуть неразрешимая ситуация взаимной блокировки. В случае взаимной блокировки возникнет ошибка Task Execution Timeout (Таймаут выполнения задачи) и работа контролера будет полностью остановлена. Ниже показан пример возникновения взаимной блокировки.



- В указанном ниже случае произойдет ошибка. Значение *Out* при этом не изменится.
  a) Предпринята попытка настроить более 16 777 215 областей блокировки одновременно.

Пример программы

В данном примере программа P1 в задаче T1 и программа P2 в задаче T2 обращаются к одной и той же глобальной переменной *GTable1*.
Когда значение *WriteReq* (запрос на запись) меняется на ИСТИНА, программа P1 записывает одну запись в массив записей GTable1.Record[] и увеличивает GTable1.Index на 1.
Когда значение *ReadReq* (запрос на чтение) меняется на ИСТИНА, программа P2 увеличивает GTable1.Index на 1 и считывает одну запись из GTable1.Record[].
Чтобы операции чтения и записи не выполнялись одновременно, используется команда Lock.

Определение глобальной переменной *GTable*

● Тип данных

Переменная	Тип данных	Комментарий
USERTABLE	STRUCT	Структура хранения записей
Index	INT	Указатель
Record	ARRAY[0..99] OF LREAL	Массив записей

● Глобальные переменные

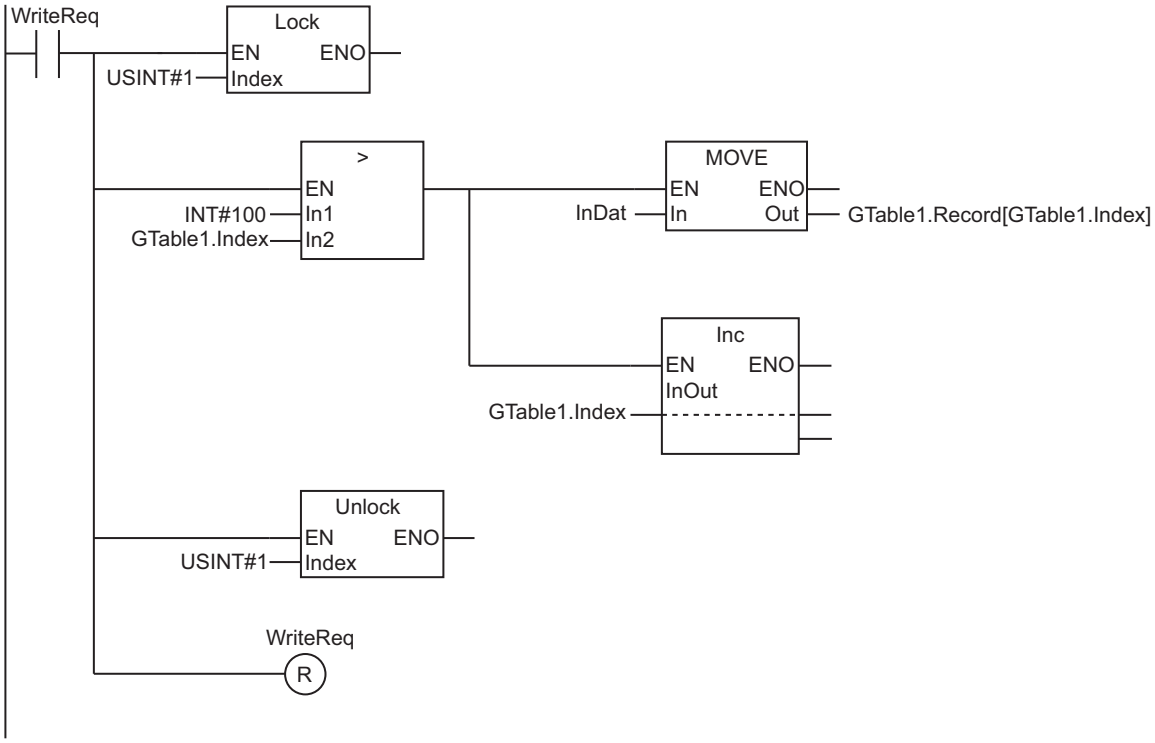
Переменная	Тип данных	Начальное значение	Комментарий
GTable1	USERTABLE	(Index:=0,Record:=[100(0.0)])	Структура хранения записей

Программа P1

● Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	WriteReq	BOOL	ЛОЖЬ	Запрос на запись
	InDat	LREAL	0,0	Записываемые данные

Внешние переменные	Переменная	Тип данных	Комментарий
	GTable1	USERTABLE	Структура хранения записей



● Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	WriteReq	BOOL	ЛОЖЬ	Запрос на запись
	InDat	LREAL	0,0	Записываемые данные

Внешние переменные	Переменная	Тип данных	Комментарий
	GTable1	USERTABLE	Структура хранения записей

```
// Обнаружение запроса на запись.
IF (WriteReq=TRUE) THEN

    // Выполнение команды Lock.
```

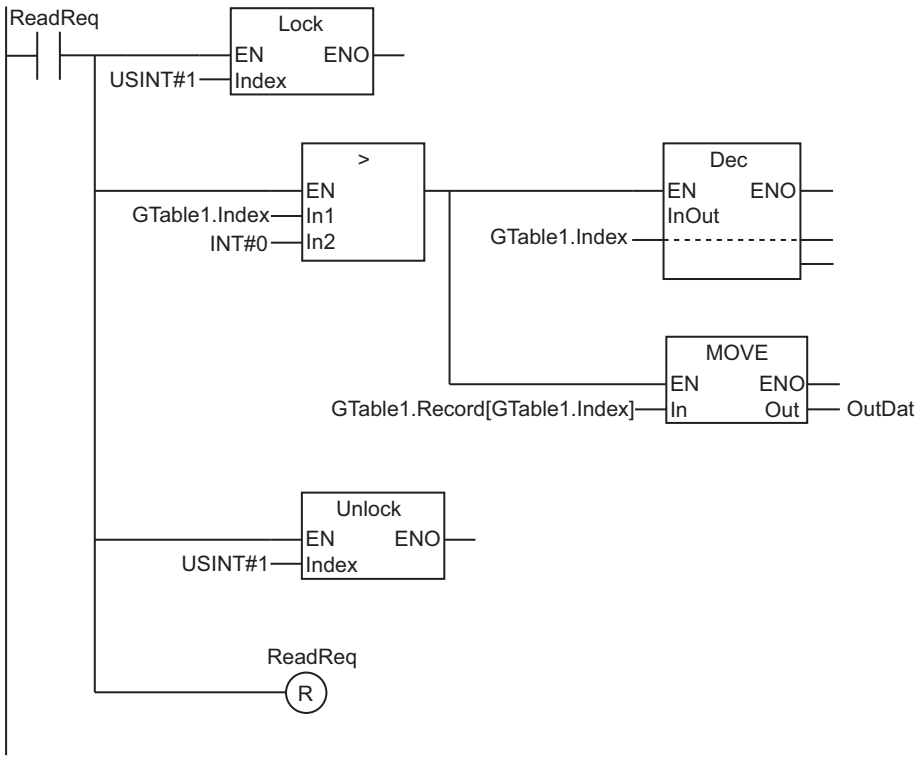
```
    Lock(USINT#1);

    IF (INT#100>GTable1.Index) THEN
        GTable1.Record[GTable1.Index]:=InDat;
        GTable1.Index                :=GTable1.Index+INT#1;
    END_IF;

    // Выполнение команды Unlock.
    Unlock(USINT#1);
    WriteReq:=FALSE;

END_IF;
```


Программа P2

● Программа на языке LD

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	ReadReq	BOOL	ЛОЖЬ	Запрос на чтение
	OutDat	LREAL	0,0	Прочитанные данные

Внешние переменные	Переменная	Тип данных	Комментарий
	GTable1	USERTABLE	Структура хранения записей



● Программа на языке ST

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	ReadReq	BOOL	ЛОЖЬ	Запрос на чтение

Внутренние переменные	Переменная	Тип данных	Начальное значение	Комментарий
	OutDat	LREAL	0,0	Прочитанные данные

Внешние переменные	Переменная	Тип данных	Комментарий
	GTable1	USERTABLE	Структура хранения записей

```
// Обнаружение запроса на чтение.
IF (ReadReq=TRUE) THEN

    // Выполнение команды Lock.
    Lock(USINT#1);

    IF (GTable1.Index>INT#0) THEN
        GTable1.Index:=GTable1.Index-INT#1;
        OutDat       :=GTable1.Record[GTable1.Index];
    END_IF;

    // Выполнение команды Unlock.
    Unlock(USINT#1);
    ReadReq:=FALSE;

END_IF;
```

ActEventTask

Команда ActEventTask инициирует выполнение событийной задачи.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
ActEventTask	Инициация событийной задачи	FUN	(@)ActEventTask EN ENO TaskName Out	ActEventTask(TaskName);

Информация о версии

Для использования этой команды требуются модуль ЦПУ с версией модуля 1.03 или более поздней и Sysmac Studio версии 1.04 или выше.

Переменные

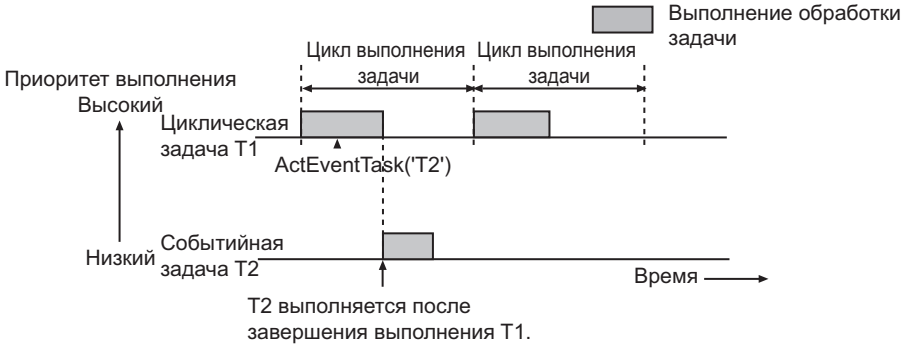
	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
TaskName	Имя задачи	Вход	Имя инициируемой событийной задачи	Макс. 64 байт (63 однобайтовых буквенно-цифровых символа + последний символ NULL)	---	"
Out	Возвращаемое значение	Выход	ИСТИНА: команда была выполнена без каких-либо ошибок. ЛОЖЬ: команда не была выполнена или произошла ошибка.	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TaskName																				OK
Out	OK																			

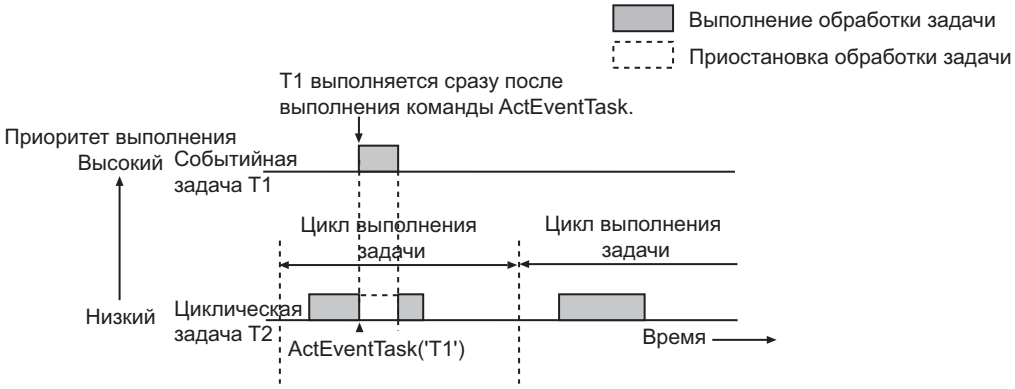
Функция

Команда ActEventTask инициирует выполнение событийной задачи с именем *TaskName*. Событийная задача выполняется в соответствии со своим приоритетом выполнения.

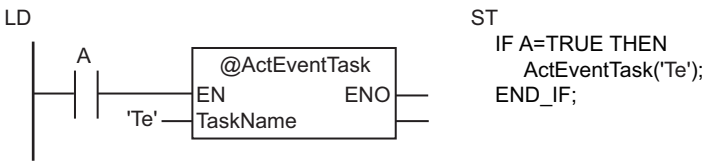
Если инициируемая событийная задача имеет более низкий приоритет выполнения, чем задача, в которой выполнена данная команда, событийная задача выполняется после завершения выполнения задачи, в которой была выполнена команда.
Пусть, например, приоритет выполнения событийной задачи T2 меньше приоритета выполнения циклической задачи T1.
Если в задаче T1 выполняется команда ActEventTask для задачи T2, выполнение задачи T1 завершается до выполнения задачи T2.



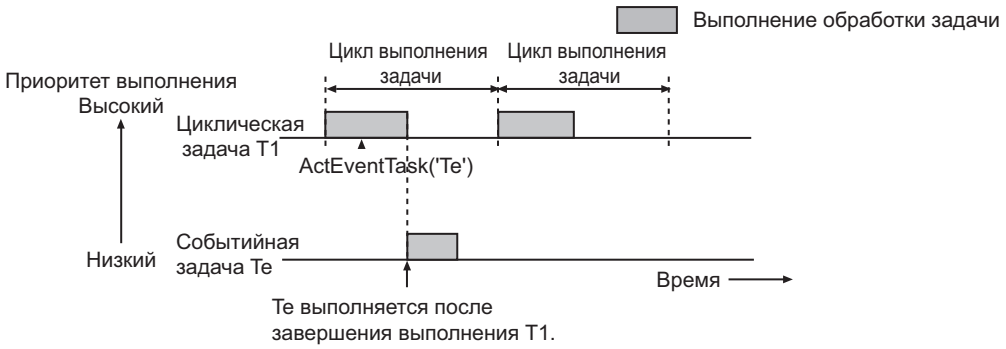
Если инициируемая событийная задача имеет более высокий приоритет выполнения, чем задача, в которой выполнена данная команда, выполнение задачи, в которой была выполнена команда, приостанавливается и выполняется событийная задача.
Пусть, например, приоритет выполнения циклической задачи T2 меньше приоритета выполнения событийной задачи T1.
Если в задаче T2 выполняется команда ActEventTask для задачи T1, выполнение задачи T2 приостанавливается для выполнения задачи T1.



Пример программы представлен на рисунке ниже. Когда значение переменной *A* = ИСТИНА, выполняется событийная задача «Te».



Предположим, что программа с этими командами назначена циклической задаче T1 и что приоритет выполнения задачи Te ниже, чем у задачи T1. Если эта команда будет выполнена в задаче T1, выполнение задачи T1 будет завершено до выполнения задачи Te.



Связанные системные переменные

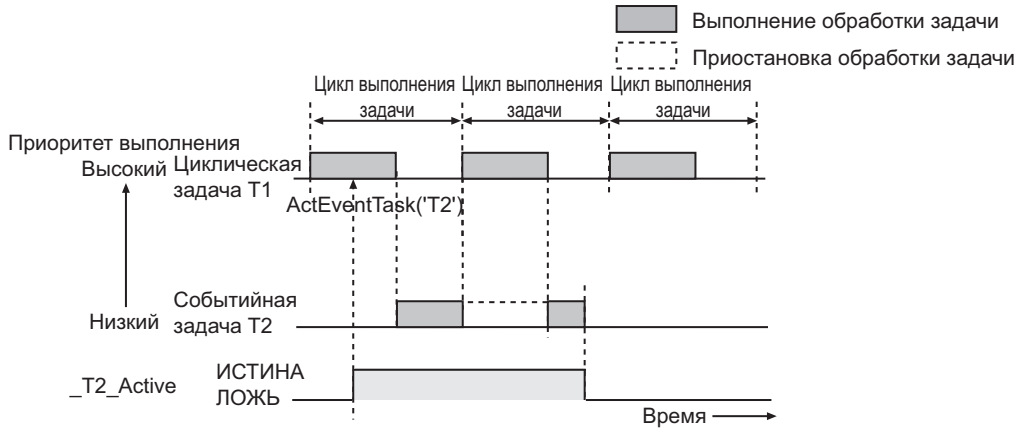
Имя	Значение	Тип данных	Описание
_**_Active *1	Флаг активности задачи	BOOL	Эта переменная указывает состояние выполнения задачи. *2 ИСТИНА: выполняется в данный момент. ЛОЖЬ: остановлена.

*1. Вместо звездочек (**) подставляется имя задачи.
*2. Дополнительные сведения см. в: *Серия NJ/NX, модули ЦПУ — Программное обеспечение. Руководство пользователя (Cat. No. W501)*.

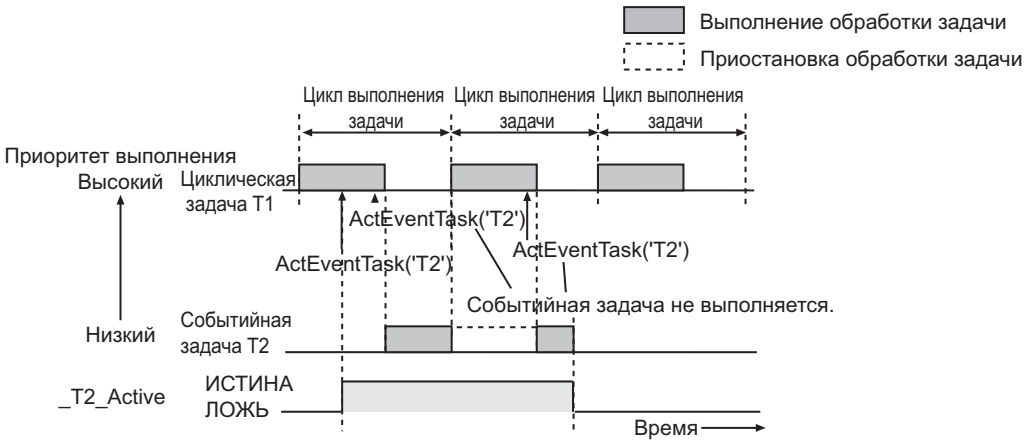
Дополнительная информация

Поведение системной переменной _**_*Active*

- При выполнении этой команды системная переменная _**_*Active* для указанной событийной задачи принимает значение ИСТИНА.
  Когда выполнение событийной задачи завершается, она снова принимает значение ЛОЖЬ.
  Пусть, например, приоритет выполнения событийной задачи T2 меньше приоритета выполнения циклической задачи T1.
  Когда в задаче T1 выполняется команда ActEventTask для задачи T2, значение системной переменной *_T2_Active* изменяется так, как показано на следующем рисунке.



- Выполнение данной команды не инициирует выполнение событийной задачи, если команда выполняется, когда системная переменная _**_*Active* содержит значение ИСТИНА.

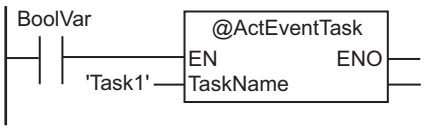


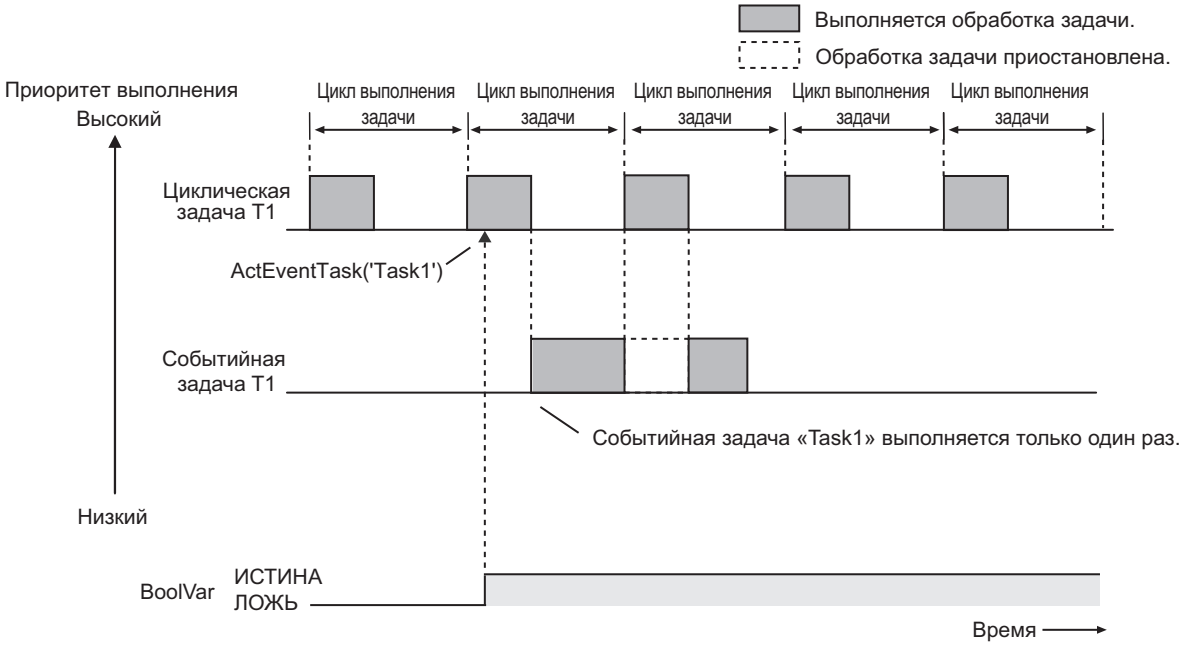
Однократное и повторяющееся выполнение событийной задачи

Ниже показаны варианты программирования для двух разных случаев. В первом случае событийная задача выполняется только один раз, когда изменяется значение указанной переменной. Во втором случае событийная задача выполняется многократно, пока переменная содержит определенное значение.

● Однократное выполнение событийной задачи при изменении значения указанной переменной

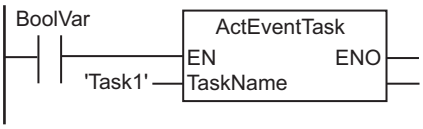
Для команды указано выполнение по положительному фронту (см. рис. ниже), поэтому событийная задача «Task1» будет выполнена только один раз, когда значение переменной *BoolVar* типа BOOL поменяется с ЛОЖЬ на ИСТИНА.

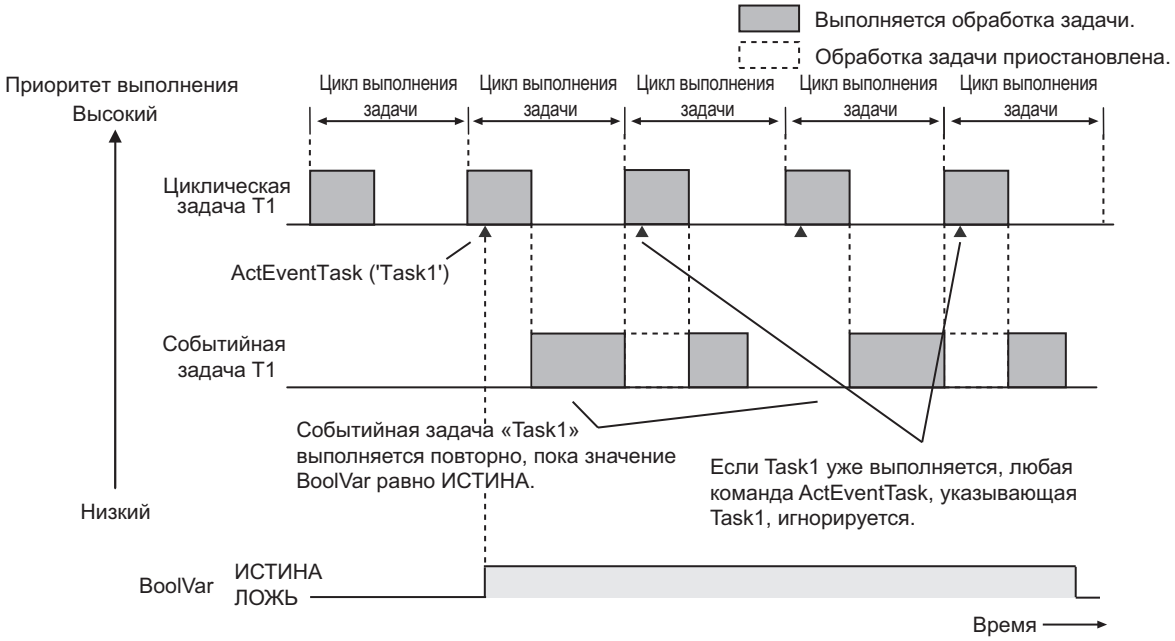




● Повторяющееся выполнение событийной задачи, пока переменная содержит указанное значение

Для команды не указано выполнение по положительному фронту (см. рис. ниже), поэтому выполнение событийной задачи «Task1» будет повторяться, пока переменная *BoolVar* типа BOOL будет содержать значение ИСТИНА. Но если эта команда будет выполнена для задачи Task1, когда задача Task1 выполняется, команда будет проигнорирована.





Меры предосторожности для обеспечения надлежащей эксплуатации

- Чтобы сократить время выполнения команд, выполняйте эту команду, только когда это необходимо для выполнения событийной задачи.
  Если эта команда выполняется, когда системная переменная _**_*Active* = ИСТИНА, для ее выполнения все равно требуется время, хотя событийная задача фактически и не запускается.
- Если событийной задачи с указанным именем *TaskName* не существует, произойдет ошибка. Выход *ENO* перейдет в состояние ЛОЖЬ.

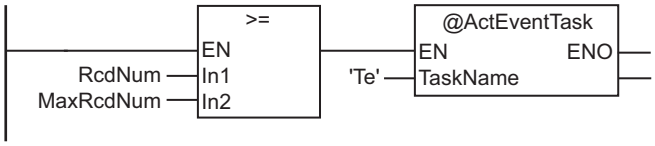
Пример программы

Пример выполнения событийной задачи, когда значение переменной удовлетворяет заданному условию

Событийная задача «Te» выполняется только один раз, когда значение переменной *RcdNum*, которое было меньше значения переменной *MaxRcdNum*, изменяется и становится больше или равно значению *MaxRcdNum*.

● Программа на языке LD

Переменная	Тип данных	Начальное значение
RcdNum	INT	0
MaxRcdNum	INT	100



● Программа на языке ST

Переменная	Тип данных	Начальное значение
RcdNum	INT	0
MaxRcdNum	INT	100
met	BOOL	ЛОЖЬ

```
IF (RcdNum>=MaxRcdNum) THEN
   IF (met=FALSE) THEN
      ActEventTask('Te');
      met:=TRUE;
   END_IF;
ELSE
   met:=FALSE;
END_IF;
```

Пример подтверждения завершения событийной задачи перед продолжением

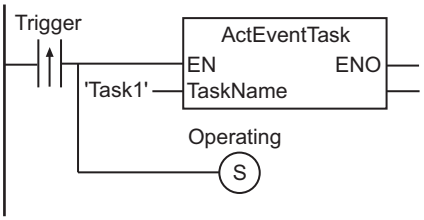
В этом примере событийная задача «Task1» выполняется каждый раз, когда переменная *Trigger* принимает значение ИСТИНА.
Используется команда Task_IsActive для проверки, завершилось ли выполнение задачи «Task 1».

● Программа на языке LD

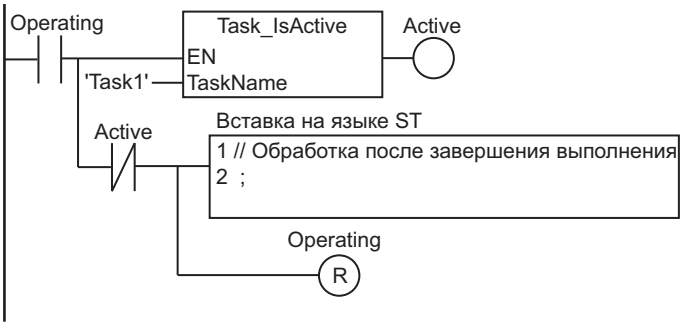
Имя	Тип данных	Начальное значение	Комментарий
Trigger	BOOL	ЛОЖЬ	Условие выполнения
Operating	BOOL	ЛОЖЬ	Проверка текущего выполнения событийной задачи

Имя	Тип данных	Начальное значение	Комментарий
Active	BOOL	ЛОЖЬ	Выполняется событийная задача

Принимается сигнал запуска (Trigger), и выполняется ActEventTask.



Task_IsActive используется, чтобы проверить, выполняется ли в данный момент Task1.



● Программа на языке ST

Имя	Тип данных	Начальное значение	Комментарий
Trigger	BOOL	ЛОЖЬ	Условие выполнения
LastTrigger	BOOL	ЛОЖЬ	Значение *Trigger* в предыдущем цикле выполнения задачи
Operating	BOOL	ЛОЖЬ	Проверка текущего выполнения событийной задачи
Active	BOOL	ЛОЖЬ	Выполняется событийная задача

```
// Запуск последовательности при переходе Trigger в состояние ИСТИНА.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) ) THEN
   ActEventTask('Task1');        // Выполнить событийную задачу «Task1».
   Operating:=TRUE;
END_IF;
LastTrigger:=Trigger;

// Посмотрим, выполняется ли задача «Task1».
IF (Operating=TRUE) THEN
   Active:=Task_IsActive('Task1');

   IF (Active=FALSE) THEN        // Выполнение задачи «Task1» завершено.
      Operating:=FALSE;
   END_IF;

END_IF;
```

Get**Clk

Команда Get**Clk выдает последовательность тактовых импульсов с заданным периодом.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Get**Clk	Группа получения тактовых импульсов	FUN	Get**Clk EN — Out "**" — следует подставить «100 μs», «1 ms», «10 ms», «20 ms», «100 ms», «1 s» или «1 min».	Out:=Get**Clk(); "**" — следует подставить «100 μs», «1 ms», «10 ms», «20 ms», «100 ms», «1 s» или «1 min».

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Тактовые импульсы	Выход	Тактовые импульсы	Зависит от типа данных.	---	---

	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	ОК																			

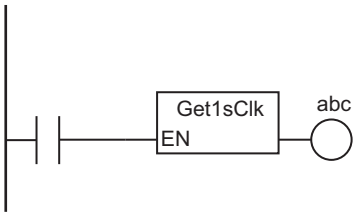
Функция

Команда Get**Clk выдает последовательность тактовых импульсов с заданным периодом. Период тактовых импульсов может составлять 100 мкс, 1 мс, 10 мс, 20 мс, 100 мс, 1 с или 1 мин.

Имя команды определяется периодом тактовых импульсов. Например, если период тактовых импульсов составляет 10 мс, команда имеет имя Get10msClk.

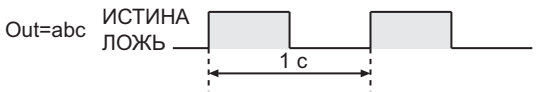
Ниже приведен пример для команды Get1sClk.

LD



ST

```
abc:=Get1sClk();
```

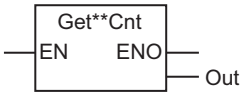


Меры предосторожности для обеспечения надлежащей эксплуатации

- Первое значение *Out* после выполнения не определено.
- При использовании этой команды в лестничной диаграмме значение выхода *Out* меняется на ЛОЖЬ, если в предыдущей логической цепи происходит ошибка.

Get**Cnt

Команда Get**Cnt возвращает текущее значение автономного счетчика заданных интервалов.

Команда	Имя	FB/ FUN	Графическое представление	Выражение языка ST
Get**Cnt	Группа получения значения автономного прямого счетчика	FUN	 "**" — следует подставить «100 ns», «1 µs», «1 ms», «10 ms», «100 ms» или «1 s».	Out:=Get**Cnt(); "**" — следует подставить «100 ns», «1 µs», «1 ms», «10 ms», «100 ms» или «1 s».

Переменные

	Значение	Вход/ выход	Описание	Диапазон допустимых значений	Единица	По умолчанию
Out	Count	Выход	Значение автономного счетчика	Зависит от типа данных.	---	---

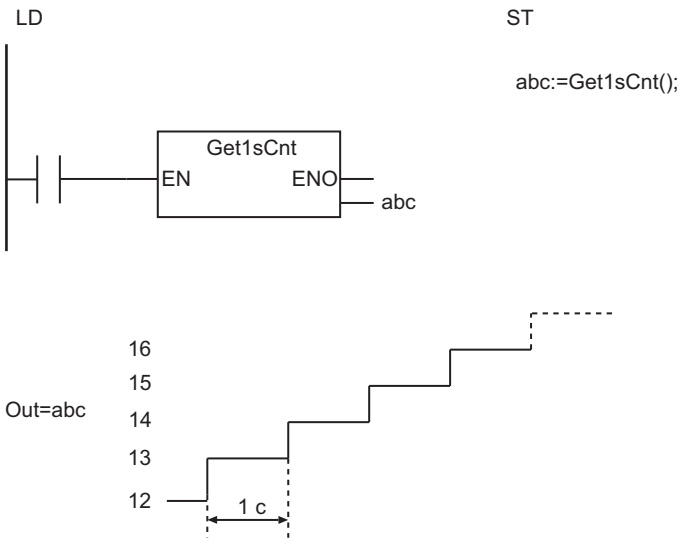
	Логический тип	Битовые строки				Целочисленные типы								Вещественные типы		Значения времени и продолжительности, даты и текстовые строки				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out									OK											

Функция

Команда Get**Cnt служит для чтения текущего значения автономного счетчика заданных интервалов.
Автономный счетчик — это счетчик, значение которого увеличивается с заданной периодичностью (т. е. счетчик подсчитывает интервалы времени). Переменная *Out* содержит текущее значение счетчика. Период приращения счетчика может составлять 100 нс, 1 мкс, 1 мс, 10 мс, 100 мс или 1 с.

Имя команды определяется периодом приращения счетчика. Например, если период приращения счетчика составляет 10 мс, команда имеет имя Get10msCnt.

Ниже приведен пример для команды Get1sCnt.



Меры предосторожности для обеспечения надлежащей эксплуатации

- Автономные счетчики начинают подсчет сразу после включения питания. Когда подсчитанное значение выходит за допустимый диапазон типа данных ULINT (18 446 744 073 709 551 615), счетчик обнуляется и счет продолжается.
- Данная команда возвращает только текущее значение автономного счетчика. Она не сбрасывает счетчик в 0.
- Начальное значение *Out* не определено. Оно не обязательно начинается с 0.

A

Приложения

A-1 Коды ошибок в переменной ErrorID

Ошибкам, которые могут возникать во время выполнения команд, присвоены определенные коды ошибок. При использовании команд, имеющих выходную переменную для вывода кода ошибки (*ErrorID*), в программе можно предусматривать обработку ошибок с использованием кодов ошибок.

В следующей таблице перечислены команды, имеющие переменную *ErrorID*, и коды ошибок, которые могут возникать при выполнении этих команд.
Значения кодов ошибок см. в документе *Серия NJ/NX — Поиск и устранение неполадок. Руководство (Cat. No. W503)*.

Дополнительная информация

Для команд, не имеющих выхода *ErrorID*, сведения об ошибках можно посмотреть в сведениях о событиях в журнале ошибок.

Тип	Команда	Имя	Код ошибки	Наименование ошибки
Команды для аналогового регулирования	PIDAT	ПИД-регулятор с автонастройкой	16#0400	Входное значение вне диапазона
			16#0401	Несоответствие входных данных
	PIDAT_HeatCool	ПИД-регулятор нагрева/охлаждения с автонастройкой	16#0400	Входное значение вне диапазона
			16#0401	Несоответствие входных данных
	AC_StepProgram	Поэтапная программа	16#0400	Входное значение вне диапазона
Команды для управления системой	ResetPLCError	Сброс ошибки контроллера в PLC	---	---
	ResetCJBError	Сброс ошибки шины ввода-вывода.	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
	ResetMCError	Сброс ошибки в Motion Control	---	---
	ResetECError	Сброс ошибки в EtherCAT	16#041A	Многократное выполнение команд
	GetNXUnitError	Получить состояние ошибки в модуле NX	16#0400	Входное значение вне диапазона
			16#041A	Многократное выполнение команд
			16#2C00	Ошибка сообщения NX
			16#2C02	Таймаут сообщения NX
	ResetUnit	Перезапустить модуль	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
			16#040F	Сбой перезапуска модуля

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	RestartNXUnit	Перезапуск модуля NX	16#0400	Входное значение вне диапазона
			16#0419	Неправильный тип данных
			16#2C00	Ошибка сообщения NX
			16#2C01	Ресурсы для сообщений NX исчерпаны
			16#2C02	Таймаут сообщения NX
			16#2C05	Ошибка сети EtherCAT для сообщения NX
			16#2C06	Внешний перезапуск уже выполнен для указанных модулей NX
			16#2C07	Для команды указан неприменимый модуль
	NX_ChangeWriteMode	Перевод модуля NX в режим записи	16#0400	Входное значение вне диапазона
			16#0419	Неправильный тип данных
			16#2C00	Ошибка сообщения NX
			16#2C01	Ресурсы для сообщений NX исчерпаны
			16#2C02	Таймаут сообщения NX
			16#2C05	Ошибка сети EtherCAT для сообщения NX
			16#2C07	Для команды указан неприменимый модуль
	NX_SaveParam	Сохранение параметров модуля NX	16#0400	Входное значение вне диапазона
			16#0419	Неправильный тип данных
			16#2C00	Ошибка сообщения NX
			16#2C01	Ресурсы для сообщений NX исчерпаны
			16#2C02	Таймаут сообщения NX

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	NX_ReadTotalPowerOnTime	Чтение общего времени работы модуля NX	16#0400	Входное значение вне диапазона
			16#0419	Неправильный тип данных
			16#2C00	Ошибка сообщения NX
			16#2C01	Ресурсы для сообщений NX исчерпаны
			16#2C02	Таймаут сообщения NX
			16#2C08	Недействительная запись общего времени наработки
Команды для обмена данными по интерфейсу EtherCAT	EC_CoESDOWrite	Запись в SDO CoE в EtherCAT	16#0400	Входное значение вне диапазона
			16#1800	Ошибка связи по сети EtherCAT
			16#1801	Ведомое устройство EtherCAT не отвечает
			16#1802	Таймаут EtherCAT
			16#1804	Ошибка прерывания SDO
			16#1808	Ресурсы связи исчерпаны
	EC_CoESDORead	Чтение из SDO CoE в EtherCAT	16#0400	Входное значение вне диапазона
			16#1800	Ошибка связи по сети EtherCAT
			16#1801	Ведомое устройство EtherCAT не отвечает
			16#1802	Таймаут EtherCAT
			16#1803	Переполнение буфера приема
			16#1804	Ошибка прерывания SDO
			16#1808	Ресурсы связи исчерпаны
	EC_StartMon	Запуск мониторинга пакетов EtherCAT	16#1805	Сохранение файла мониторинга пакетов
			16#1807	Функция мониторинга пакетов работает
			16#1808	Ресурсы связи исчерпаны
			16#1809	Функция мониторинга пакетов не поддерживается

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	EC_StopMon	Остановка мониторинга пакетов EtherCAT	16#1806	Функция мониторинга пакетов не запущена
			16#1808	Ресурсы связи исчерпаны
			16#1809	Функция мониторинга пакетов не поддерживается
	EC_SaveMon	Сохранение пакетов EtherCAT	16#1805	Сохранение файла мониторинга пакетов
			16#1807	Функция мониторинга пакетов работает
			16#1808	Ресурсы связи исчерпаны
			16#1809	Функция мониторинга пакетов не поддерживается
	EC_CopyMon	Передача пакетов EtherCAT	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1402	Недостаточная емкость карты памяти SD
			16#1403	Файл не существует
			16#1404	Слишком много файлов/каталогов
			16#1405	Файл уже используется
			16#140A	Отказано в доступе для записи
			16#140B	Открыто слишком много файлов
			16#140D	Слишком длинное имя файла или каталога
			16#140E	Доступ к карте памяти SD завершился сбоем
			16#1808	Ресурсы связи исчерпаны
			16#1809	Функция мониторинга пакетов не поддерживается

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	EC_DisconnectSlave	Отсоединение ведомого устройства EtherCAT	16#1800	Ошибка связи по сети EtherCAT
			16#1801	Ведомое устройство EtherCAT не отвечает
			16#1808	Ресурсы связи исчерпаны
			16#180A	Невозможно выполнить команду для целевого ведомого устройства
	EC_ConnectSlave	Подсоединение ведомого устройства EtherCAT	16#1800	Ошибка связи по сети EtherCAT
			16#1801	Ведомое устройство EtherCAT не отвечает
			16#1808	Ресурсы связи исчерпаны
			16#180A	Невозможно выполнить команду для целевого ведомого устройства
	EC_ChangeEnableSetting	Активация/деактивация ведомого устройства EtherCAT	16#1800	Ошибка связи по сети EtherCAT
			16#1801	Ведомое устройство EtherCAT не отвечает
			16#1808	Ресурсы связи исчерпаны
			16#180A	Невозможно выполнить команду для целевого ведомого устройства
	NX_WriteObj	Запись в объект модуля NX	16#0400	Входное значение вне диапазона
			16#0419	Неправильный тип данных
			16#041B	Превышен объем данных
			16#2C00	Ошибка сообщения NX
			16#2C01	Ресурсы для сообщений NX исчерпаны
			16#2C02	Таймаут сообщения NX
			16#2C03	Неверная длина сообщения NX

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	NX_ReadObj	Чтение объекта модуля NX	16#0400	Входное значение вне диапазона
			16#0410	Ошибка формата текстовой строки
			16#0419	Неправильный тип данных
			16#041C	Разные размеры данных
			16#2C00	Ошибка сообщения NX
			16#2C01	Ресурсы для сообщений NX исчерпаны
			16#2C02	Таймаут сообщения NX
Команды для обмена данными по интерфейсу IO-Link	IOL_ReadObj	Чтение из объекта устройства IO-Link	16#0400	Входное значение вне диапазона
			16#0410	Ошибка формата текстовой строки
			16#0419	Неправильный тип данных
			16#041C	Разные размеры данных
			16#4800	Принята ошибка от устройства
			16#4801	Указанного модуля не существует
			16#4802	Превышен лимит обработки сообщений
			16#4803	Ошибка состояния указанного модуля
			16#4804	Слишком много одновременно выполняемых команд
			16#4805	Таймаут связи
			16#4806	Недопустимый режим
			16#4807	Состояние выключенного питания ввода-вывода
			16#4808	Ошибка проверки

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	IOL_WriteObj	Запись в объект устройства IO-Link	16#0400	Входное значение вне диапазона
			16#0419	Неправильный тип данных
			16#041B	Превышен объем данных
			16#4800	Принята ошибка от устройства
			16#4801	Указанного модуля не существует
			16#4802	Превышен лимит обработки сообщений
			16#4803	Ошибка состояния указанного модуля
			16#4804	Слишком много одновременно выполняемых команд
			16#4805	Таймаут связи
			16#4806	Недопустимый режим
			16#4807	Состояние выключенного питания ввода-вывода
			16#4808	Ошибка проверки
Команды для обмена данными по интерфейсу EtherNet/IP	CIPOpen	Открытие CIP-соединения класса 3 (Large_Forward_Open)	16#0400	Входное значение вне диапазона
			16#1C00	Ошибка явного сообщения
			16#1C01	Неверный путь маршрута
			16#1C03	Ресурсы связи по протоколу CIP исчерпаны
			16#1C04	Таймаут CIP
			16#1C05	Соединение класса 3 не установлено
			16#2000	Ошибка настройки локального IP-адреса
			16#2004	Локальный IP-адрес не задан

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	CIPOpenWithDataSize	Открытие CIP-соединения класса 3 с указанным объемом данных	16#0400	Входное значение вне диапазона
			16#1C00	Ошибка явного сообщения
			16#1C01	Неверный путь маршрута
			16#1C03	Ресурсы связи по протоколу CIP исчерпаны
			16#1C04	Таймаут CIP
			16#1C05	Соединение класса 3 не установлено
			16#2000	Ошибка настройки локального IP-адреса
			16#2004	Локальный IP-адрес не задан
	CIPRead	Чтение переменной посредством явного сообщения класса 3	16#0400	Входное значение вне диапазона
			16#0407	Превышен диапазон данных
			16#0419	Неправильный тип данных
			16#1C00	Ошибка явного сообщения
			16#1C02	Дескриптор CIP вне диапазона
			16#1C03	Ресурсы связи по протоколу CIP исчерпаны
			16#1C04	Таймаут CIP
			16#1C06	Превышен объем данных для протокола CIP
	CIPWrite	Запись переменной посредством явного сообщения класса 3	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0407	Превышен диапазон данных
			16#0419	Неправильный тип данных
			16#1C00	Ошибка явного сообщения
			16#1C02	Дескриптор CIP вне диапазона
			16#1C03	Ресурсы связи по протоколу CIP исчерпаны
			16#1C04	Таймаут CIP
			16#1C06	Превышен объем данных для протокола CIP

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	CIPSend	Передача явного сообщения класса 3	16#0400	Входное значение вне диапазона
			16#0401	Несоответствие входных данных
			16#0406	Указано недопустимое расположение данных
			16#0407	Превышен диапазон данных
			16#0419	Неправильный тип данных
			16#1C00	Ошибка явного сообщения
			16#1C02	Дескриптор CIP вне диапазона
			16#1C03	Ресурсы связи по протоколу CIP исчерпаны
			16#1C04	Таймаут CIP
			16#1C06	Превышен объем данных для протокола CIP
	CIPClose	Закрытие CIP-соединения класса 3	16#1C02	Дескриптор CIP вне диапазона
			16#1C03	Ресурсы связи по протоколу CIP исчерпаны
	CIPUCMMRead	Чтение переменной посредством явного сообщения UCMM	16#0400	Входное значение вне диапазона
			16#0407	Превышен диапазон данных
			16#0419	Неправильный тип данных
			16#1C00	Ошибка явного сообщения
			16#1C01	Неверный путь маршрута
			16#1C03	Ресурсы связи по протоколу CIP исчерпаны
			16#1C04	Таймаут CIP
			16#2000	Ошибка настройки локального IP-адреса
			16#2004	Локальный IP-адрес не задан

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	CIPUCMMWrite	Запись переменной посредством явного сообщения UCMM	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0419	Неправильный тип данных
			16#1C00	Ошибка явного сообщения
			16#1C01	Неверный путь маршрута
			16#1C03	Ресурсы связи по протоколу CIP исчерпаны
			16#1C04	Таймаут CIP
			16#2000	Ошибка настройки локального IP-адреса
			16#2004	Локальный IP-адрес не задан
	CIPUCMMSend	Передача явного сообщения UCMM	16#0400	Входное значение вне диапазона
			16#0401	Несоответствие входных данных
			16#0406	Указано недопустимое расположение данных
			16#0407	Превышен диапазон данных
			16#0419	Неправильный тип данных
			16#1C00	Ошибка явного сообщения
			16#1C01	Неверный путь маршрута
			16#1C03	Ресурсы связи по протоколу CIP исчерпаны
			16#1C04	Таймаут CIP
			16#2000	Ошибка настройки локального IP-адреса
			16#2004	Локальный IP-адрес не задан

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	SktUDPCreate	Создание сокета UDP	16#0400	Входное значение вне диапазона
			16#2000	Ошибка настройки локального IP-адреса
			16#2001	Порт TCP/UDP уже используется
			16#2003	Ошибка состояния сокета
			16#2004	Локальный IP-адрес не задан
			16#2008	Ресурсы связи через сокеты исчерпаны
	SktUDPRcv	Прием через сокет UDP	16#0400	Входное значение вне диапазона
			16#0407	Превышен диапазон данных
			16#0419	Неправильный тип данных
			16#2003	Ошибка состояния сокета
			16#2006	Таймаут сокета
			16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны
	SktUDPSend	Передача через сокет UDP	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0419	Неправильный тип данных
			16#2002	Не удалось разрешить адрес
			16#2003	Ошибка состояния сокета
			16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	SktTCPAccept	Принятие сокета TCP	16#0400	Входное значение вне диапазона
			16#2000	Ошибка настройки локального IP-адреса
			16#2001	Порт TCP/UDP уже используется
			16#2002	Не удалось разрешить адрес
			16#2003	Ошибка состояния сокета
			16#2004	Локальный IP-адрес не задан
			16#2006	Таймаут сокета
			16#2008	Ресурсы связи через сокеты исчерпаны
	SktTCPConnect	Соединение с сокетом TCP	16#0400	Входное значение вне диапазона
			16#2000	Ошибка настройки локального IP-адреса
			16#2001	Порт TCP/UDP уже используется
			16#2002	Не удалось разрешить адрес
			16#2003	Ошибка состояния сокета
			16#2004	Локальный IP-адрес не задан
			16#2005	Невозможно использовать встроенный порт EtherNet/IP
			16#2006	Таймаут сокета
			16#2008	Ресурсы связи через сокеты исчерпаны
	SktTCPRcv	Прием через сокет TCP	16#0400	Входное значение вне диапазона
			16#0407	Превышен диапазон данных
			16#0419	Неправильный тип данных
			16#2003	Ошибка состояния сокета
			16#2006	Таймаут сокета
			16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	SktTCPSend	Передача через сокет TCP	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0419	Неправильный тип данных
			16#2003	Ошибка состояния сокета
			16#2006	Таймаут сокета
			16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны
	SktGetTCPStatus	Чтение состояния сокета TCP	16#2003	Ошибка состояния сокета
			16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны
	SktClose	Закрытие сокета TCP/UDP	16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны
	SktClearBuf	Очистка буфера приема сокета TCP/UDP	16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны
	SktSetOption	Настройка дополнительного параметра сокета TCP	16#0400	Входное значение вне диапазона
			16#0419	Неправильный тип данных
			16#2003	Ошибка состояния сокета
			16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	ModbusTCPCmd	Передача команды общего назначения по протоколу Modbus TCP	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0407	Превышен диапазон данных
			16#0C10	Исключение ответа Modbus
			16#0C11	Недопустимый ответ Modbus
			16#2003	Ошибка состояния сокета
			16#2006	Таймаут сокета
			16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны
	ModbusTCPRead	Передача команды чтения по протоколу Modbus TCP	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0407	Превышен диапазон данных
			16#0419	Неправильный тип данных
			16#0C10	Исключение ответа Modbus
			16#0C11	Недопустимый ответ Modbus
			16#2003	Ошибка состояния сокета
			16#2006	Таймаут сокета
			16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	ModbusTCPWrite	Передача команды записи по протоколу Modbus TCP	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0419	Неправильный тип данных
			16#0C10	Исключение ответа Modbus
			16#0C11	Недопустимый ответ Modbus
			16#2003	Ошибка состояния сокета
			16#2006	Таймаут сокета
			16#2007	Дескриптор сокета вне диапазона
			16#2008	Ресурсы связи через сокеты исчерпаны
	ChangeIPAdr	Изменение IP-адреса	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
			16#2400	Нет права на выполнение
	ChangeFTPAccount	Изменение учетной записи FTP	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
			16#2400	Нет права на выполнение
	ChangeNTPServerAdr	Изменение адреса сервера NTP	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
			16#2400	Нет права на выполнение
	FTPGetFileList	Получение списка файлов на сервере FTP	16#0400	Входное значение вне диапазона
			16#2403	Исчерпан лимит команд для клиента FTP
			16#2405	Каталога не существует (FTP)
			16#2406	Ошибка соединения с сервером FTP
			16#2407	Сбой выполнения целевого сервера FTP

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	FTPGetFile	Получить файл с сервера FTP	16#0400	Входное значение вне диапазона
			16#2403	Исчерпан лимит команд для клиента FTP
			16#2404	Исчерпан лимит на количество файлов
			16#2405	Каталога не существует (FTP)
			16#2406	Ошибка соединения с сервером FTP
			16#2407	Сбой выполнения целевого сервера FTP
			16#2408	Сбой доступа к карте памяти SD для FTP
			16#2409	Указанный файл не существует
			16#240A	Указанный файл защищен от записи
			16#240C	Не удалось получить доступ к указанному файлу
	FTPPutFile	Поместить файл на сервер FTP	16#0400	Входное значение вне диапазона
			16#2403	Исчерпан лимит команд для клиента FTP
			16#2404	Исчерпан лимит на количество файлов
			16#2405	Каталога не существует (FTP)
			16#2406	Ошибка соединения с сервером FTP
			16#2407	Сбой выполнения целевого сервера FTP
			16#2408	Сбой доступа к карте памяти SD для FTP
			16#2409	Указанный файл не существует
			16#240A	Указанный файл защищен от записи
			16#240B	Не удалось удалить указанный файл
			16#240C	Не удалось получить доступ к указанному файлу

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	FTPRemoveFile	Удаление файла с сервера FTP	16#0400	Входное значение вне диапазона
			16#2403	Исчерпан лимит команд для клиента FTP
			16#2404	Исчерпан лимит на количество файлов
			16#2405	Каталога не существует (FTP)
			16#2406	Ошибка соединения с сервером FTP
			16#2407	Сбой выполнения целевого сервера FTP
			16#2409	Указанный файл не существует
	FTPRemoveDir	Удаление каталога с сервера FTP	16#0400	Входное значение вне диапазона
			16#2405	Каталога не существует (FTP)
			16#2406	Ошибка соединения с сервером FTP
			16#2407	Сбой выполнения целевого сервера FTP
Команды для обмена данными по последовательному интерфейсу	ExecPMCR	Макрос протокола	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0407	Превышен диапазон данных
			16#040D	Указан недопустимый модуль
			16#0413	Неопределенный адрес памяти устройства серии CJ
			16#0419	Неправильный тип данных
			16#0C00	Недопустимый режим связи по последовательному интерфейсу
			16#0800	Ошибка FINS
			16#0801	Порт FINS уже используется

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	SerialSend	Передача через последовательный порт SCU	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#0C00	Недопустимый режим связи по последовательному интерфейсу
			16#0800	Ошибка FINS
			16#0801	Порт FINS уже используется
	SerialRcv	Прием через последовательный порт SCU	16#0400	Входное значение вне диапазона
			16#0407	Превышен диапазон данных
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#0C00	Недопустимый режим связи по последовательному интерфейсу
			16#0800	Ошибка FINS
			16#0801	Порт FINS уже используется
	SerialRcvNoClear	Прием через последовательный порт SCU без очистки буфера приема	16#0400	Входное значение вне диапазона
			16#0407	Превышен диапазон данных
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#0C00	Недопустимый режим связи по последовательному интерфейсу
			16#0800	Ошибка FINS
			16#0801	Порт FINS уже используется

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	SendCmd	Передача команды	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0407	Превышен диапазон данных
			16#0419	Неправильный тип данных
			16#0800	Ошибка FINS
			16#0801	Порт FINS уже используется
	NX_SerialSend	Передача данных в беспротокольном режиме	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C04	Одновременный доступ к портам
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	NX_SerialRcv	Прием данных в беспротокольном режиме	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0407	Превышен диапазон данных
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C03	Переполнение буфера приема
			16#0C04	Одновременный доступ к портам
			16#0C05	Ошибка проверки четности
			16#0C06	Ошибка кадра
			16#0C07	Ошибка переполнения
			16#0C0B	Таймаут последовательной связи
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	NX_ModbusRtuCmd	Передача команды общего назначения по протоколу Modbus RTU	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#0407	Превышен диапазон данных
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C03	Переполнение буфера приема
			16#0C04	Одновременный доступ к портам
			16#0C05	Ошибка проверки четности
			16#0C06	Ошибка кадра
			16#0C07	Ошибка переполнения
			16#0C08	Несоответствие контрольной суммы
			16#0C0B	Таймаут последовательной связи
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован
			16#0C10	Исключение ответа Modbus
			16#0C11	Недопустимый ответ Modbus

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	NX_ModbusRtuRead	Передача команды чтения по протоколу Modbus RTU	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C03	Переполнение буфера приема
			16#0C04	Одновременный доступ к портам
			16#0C05	Ошибка проверки четности
			16#0C06	Ошибка кадра
			16#0C07	Ошибка переполнения
			16#0C08	Несоответствие контрольной суммы
			16#0C0B	Таймаут последовательной связи
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован
			16#0C10	Исключение ответа Modbus
			16#0C11	Недопустимый ответ Modbus

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	NX_ModbusRtuWrite	Передача команды записи по протоколу Modbus RTU	16#0400	Входное значение вне диапазона
			16#0406	Указано недопустимое расположение данных
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C03	Переполнение буфера приема
			16#0C04	Одновременный доступ к портам
			16#0C05	Ошибка проверки четности
			16#0C06	Ошибка кадра
			16#0C07	Ошибка переполнения
			16#0C08	Несоответствие контрольной суммы
			16#0C0B	Таймаут последовательной связи
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован
			16#0C10	Исключение ответа Modbus
			16#0C11	Недопустимый ответ Modbus
	NX_SerialSigCtl	Включение/выключение сигнала управления последовательного порта	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C04	Одновременный доступ к портам
			16#0C0B	Таймаут последовательной связи
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	NX_SerialSigRead	Чтение сигнала управления последовательного порта	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C04	Одновременный доступ к портам
			16#0C0B	Таймаут последовательной связи
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован
	NX_SerialStatusRead	Чтение состояния последовательного порта	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C04	Одновременный доступ к портам
			16#0C0B	Таймаут последовательной связи
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	NX_SerialBufClear	Очистка буфера	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C04	Одновременный доступ к портам
			16#0C0B	Таймаут последовательной связи
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован
	NX_SerialStartMon	Запуск мониторинга линии последовательного интерфейса	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C04	Одновременный доступ к портам
			16#0C0B	Таймаут последовательной связи
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	NX_SerialStopMon	Остановка мониторинга линии последовательного интерфейса	16#0400	Входное значение вне диапазона
			16#040D	Указан недопустимый модуль
			16#0419	Неправильный тип данных
			16#041D	Превышены ресурсы для одновременного выполнения команд
			16#0C04	Одновременный доступ к портам
			16#0C0B	Таймаут последовательной связи
			16#0C0C	Команда выполнена для неприменимого порта
			16#0C0D	Модуль CIF инициализирован
Команды для работы с картой памяти SD	FileWriteVar	Запись переменной в файл	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1402	Недостаточная емкость карты памяти SD
			16#1403	Файл не существует
			16#1404	Слишком много файлов/каталогов
			16#1405	Файл уже используется
			16#1409	Это имя файла уже существует
			16#140A	Отказано в доступе для записи
			16#140B	Открыто слишком много файлов
			16#140D	Слишком длинное имя файла или каталога
			16#140E	Доступ к карте памяти SD завершился сбоем

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	FileReadVar	Чтение переменной из файла	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1403	Файл не существует
			16#1405	Файл уже используется
			16#140B	Открыто слишком много файлов
			16#140D	Слишком длинное имя файла или каталога
			16#140E	Доступ к карте памяти SD завершился сбоем
	FileOpen	Открытие файла	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1403	Файл не существует
			16#1404	Слишком много файлов/каталогов
			16#1405	Файл уже используется
			16#140A	Отказано в доступе для записи
			16#140B	Открыто слишком много файлов
			16#140D	Слишком длинное имя файла или каталога
			16#140E	Доступ к карте памяти SD завершился сбоем
	FileClose	Закрытие файла	16#1400	Сбой доступа к карте памяти SD
			16#1403	Файл не существует
			16#1405	Файл уже используется
			16#140E	Доступ к карте памяти SD завершился сбоем

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	FileSeek	Поиск в файле	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1403	Файл не существует
			16#1405	Файл уже используется
			16#1407	Смещение вне диапазона
			16#140E	Доступ к карте памяти SD завершился сбоем
	FileRead	Чтение из файла	16#0406	Указано недопустимое расположение данных
			16#0419	Неправильный тип данных
			16#1400	Сбой доступа к карте памяти SD
			16#1403	Файл не существует
			16#1405	Файл уже используется
			16#1406	Несоответствие режима открытия
			16#140E	Доступ к карте памяти SD завершился сбоем
	FileWrite	Запись в файл	16#0406	Указано недопустимое расположение данных
			16#0419	Неправильный тип данных
			16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1402	Недостаточная емкость карты памяти SD
			16#1403	Файл не существует
			16#1405	Файл уже используется
			16#1406	Несоответствие режима открытия
			16#140E	Доступ к карте памяти SD завершился сбоем

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	FileGets	Получить текстовую строку	16#1400	Сбой доступа к карте памяти SD
			16#1403	Файл не существует
			16#1405	Файл уже используется
			16#1406	Несоответствие режима открытия
			16#140E	Доступ к карте памяти SD завершился сбоем
	FilePuts	Поместить текстовую строку	16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1402	Недостаточная емкость карты памяти SD
			16#1403	Файл не существует
			16#1405	Файл уже используется
			16#1406	Несоответствие режима открытия
			16#140E	Доступ к карте памяти SD завершился сбоем

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	FileCopy	Копирование файла	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1402	Недостаточная емкость карты памяти SD
			16#1403	Файл не существует
			16#1404	Слишком много файлов/каталогов
			16#1405	Файл уже используется
			16#1409	Это имя файла уже существует
			16#140A	Отказано в доступе для записи
			16#140B	Открыто слишком много файлов
			16#140D	Слишком длинное имя файла или каталога
			16#140E	Доступ к карте памяти SD завершился сбоем
	FileRemove	Удаление файла	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1403	Файл не существует
			16#1405	Файл уже используется
			16#140A	Отказано в доступе для записи
			16#140B	Открыто слишком много файлов
			16#140D	Слишком длинное имя файла или каталога
			16#140E	Доступ к карте памяти SD завершился сбоем

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	FileRename	Изменение имени файла	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1403	Файл не существует
			16#1404	Слишком много файлов/каталогов
			16#1405	Файл уже используется
			16#1408	Каталог не пуст
			16#1409	Это имя файла уже существует
			16#140A	Отказано в доступе для записи
			16#140B	Открыто слишком много файлов
			16#140D	Слишком длинное имя файла или каталога
			16#140E	Доступ к карте памяти SD завершился сбоем
	DirCreate	Создание каталога	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1402	Недостаточная емкость карты памяти SD
			16#1404	Слишком много файлов/каталогов
			16#1405	Файл уже используется
			16#1409	Это имя файла уже существует
			16#140B	Открыто слишком много файлов
			16#140C	Каталог не существует
			16#140D	Слишком длинное имя файла или каталога
			16#140E	Доступ к карте памяти SD завершился сбоем

Тип	Команда	Имя	Код ошибки	Наименование ошибки
	DirRemove	Удаление каталога	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1405	Файл уже используется
			16#1408	Каталог не пуст
			16#140A	Отказано в доступе для записи
			16#140B	Открыто слишком много файлов
			16#140C	Каталог не существует
			16#140D	Слишком длинное имя файла или каталога
			16#140E	Доступ к карте памяти SD завершился сбоем
	BackupToMemoryCard	Резервное копирование на карту памяти SD	16#0400	Входное значение вне диапазона
			16#1400	Сбой доступа к карте памяти SD
			16#1401	Карта памяти SD защищена от записи
			16#1402	Недостаточная емкость карты памяти SD
			16#1404	Слишком много файлов/каталогов
			16#1409	Это имя файла уже существует
			16#140C	Каталог не существует
			16#140E	Доступ к карте памяти SD завершился сбоем
			16#140F	Операция резервного копирования уже выполняется
			16#1410	Не удается выполнить резервное копирование
			16#1411	Сбой резервного копирования модуля/ведомого устройства

A-2 Коды ошибок

Четыре младших разряда кода события содержат код ошибки для команды.
Описание кодов ошибок см. в описании соответствующих кодов событий. Например, если для команды указан код ошибки 16#0400, обратитесь к описанию события с кодом события 54010400 hex.

Сведения о кодах событий см. в документе *Серия NJ/NX — Поиск и устранение неполадок. Руководство (Cat. No. W503)*.

Информация о версии

Коды событий для команд поддерживаются модулями ЦПУ с версией модуля 1.02 или более поздней версией.

A-3 Команды, не поддерживаемые в событийных задачах

Событийная задача выполняется только один раз, когда соблюдается указанное условие. Такие задачи не выполняются повторно (циклически) в каждом периоде (цикле) выполнения задач. Следовательно, программы, которые содержат команды, выполняемые более чем в одном цикле выполнения задачи, не могут назначаться событийным задачам.
В следующей таблице перечислены команды, выполнение которых занимает более одного цикла. Эти команды нельзя использовать в программах, назначаемых событийной задаче. Если это будет сделано, произойдет ошибка сборки.

Тип	Команда	Имя	Стр.
Команды для работы со стеком и таблицами	RecSort	Сортировка записей	стр. 2-594
Команды для аналогового регулирования	PIDAT	ПИД-регулятор с автонастройкой	стр. 2-762
	PIDAT_HeatCool	ПИД-регулятор нагрева/охлаждения с автонастройкой	стр. 2-796
	AC_StepProgram	Поэтапная программа	стр. 2-894
Команды для управления системой	ResetPLCError	Сброс ошибки контроллера в PLC	стр. 2-945
	ResetCJBError	Сброс ошибки шины ввода-вывода.	стр. 2-951
	ResetMCError	Сброс ошибки в Motion Control	стр. 2-957
	ResetECError	Сброс ошибки в EtherCAT	стр. 2-966
	ResetNXBError	Сброс ошибки в NX Bus	стр. 2-971
	GetNXUnitError	Получить состояние ошибки в модуле NX	стр. 2-975
	ResetUnit	Перезапустить модуль	стр. 2-985
	RestartNXUnit	Перезапуск модуля NX	стр. 2-992
	NX_ChangeWriteMode	Перевод модуля NX в режим записи	стр. 2-998
	NX_SaveParam	Сохранение параметров модуля NX	стр. 2-1004
	NX_ReadTotalPowerOnTime	Чтение общего времени работы модуля NX	стр. 2-1013
Команды для обмена данными по интерфейсу EtherCAT	EC_CoESDOWrite	Запись в SDO CoE в EtherCAT	стр. 2-1060
	EC_CoESDORead	Чтение из SDO CoE в EtherCAT	стр. 2-1064
	EC_StartMon	Запуск мониторинга пакетов EtherCAT	стр. 2-1070
	EC_StopMon	Остановка мониторинга пакетов EtherCAT	стр. 2-1076
	EC_SaveMon	Сохранение пакетов EtherCAT	стр. 2-1078
	EC_CopyMon	Передача пакетов EtherCAT	стр. 2-1080
	EC_DisconnectSlave	Отсоединение ведомого устройства EtherCAT	стр. 2-1082
	EC_ConnectSlave	Подсоединение ведомого устройства EtherCAT	стр. 2-1091
	EC_ChangeEnableSetting	Активация/деактивация ведомого устройства EtherCAT	стр. 2-1093
	NX_WriteObj	Запись в объект модуля NX	стр. 2-1114
	NX_ReadObj	Чтение объекта модуля NX	стр. 2-1131

Тип	Команда	Имя	Стр.
Команды для обмена данными по интерфейсу IO-Link	IOL_ReadObj	Чтение из объекта устройства IO-Link	стр. 2-1142
	IOL_WriteObj	Запись в объект устройства IO-Link	стр. 2-1151
Команды для обмена данными по интерфейсу EtherNet/IP	CIPOpen	Открытие CIP-соединения класса 3 (Large_Forward_Open)	стр. 2-1163
	CIPOpenWithDataSize	Открытие CIP-соединения класса 3 с указанным объемом данных	стр. 2-1174
	CIPRead	Чтение переменной посредством явного сообщения класса 3	стр. 2-1178
	CIPWrite	Запись переменной посредством явного сообщения класса 3	стр. 2-1184
	CIPSend	Передача явного сообщения класса 3	стр. 2-1191
	CIPClose	Закрытие CIP-соединения класса 3	стр. 2-1197
	CIPUCMMRead	Чтение переменной посредством явного сообщения UCMM	стр. 2-1200
	CIPUCMMWrite	Запись переменной посредством явного сообщения UCMM	стр. 2-1206
	CIPUCMMSend	Передача явного сообщения UCMM	стр. 2-1214
	SktUDPCreate	Создание сокета UDP	стр. 2-1227
	SktUDPRcv	Прием через сокет UDP	стр. 2-1236
	SktUDPSend	Передача через сокет UDP	стр. 2-1240
	SktTCPAccept	Принятие сокета TCP	стр. 2-1243
	SktTCPConnect	Соединение с сокетом TCP	стр. 2-1246
	SktTCPRcv	Прием через сокет TCP	стр. 2-1255
	SktTCPSend	Передача через сокет TCP	стр. 2-1259
	SktGetTCPStatus	Чтение состояния сокета TCP	стр. 2-1262
	SktClose	Закрытие сокета TCP/UDP	стр. 2-1265
	SktClearBuf	Очистка буфера приема сокета TCP/UDP	стр. 2-1268
	SktSetOption	Настройка дополнительного параметра сокета TCP	стр. 2-1271
	ChangeIPAdr	Изменение IP-адреса	стр. 2-1303
	ChangeFTPAccount	Изменение учетной записи FTP	стр. 2-1312
	ChangeNTPServerAdr	Изменение адреса сервера NTP	стр. 2-1316
	FTPGetFileList	Получение списка файлов на сервере FTP	стр. 2-1321
	FTPGetFile	Получить файл с сервера FTP	стр. 2-1337
	FTPPutFile	Поместить файл на сервер FTP	стр. 2-1347
	FTPRemoveFile	Удаление файла с сервера FTP	стр. 2-1358
	FTPRemoveDir	Удаление каталога с сервера FTP	стр. 2-1369
Команды для обмена данными по последовательному интерфейсу	ExecPMCR	Макрос протокола	стр. 2-1376
	SerialSend	Передача через последовательный порт SCU	стр. 2-1391
	SerialRcv	Прием через последовательный порт SCU	стр. 2-1403
	SerialRcvNoClear	Прием через последовательный порт SCU без очистки буфера приема	стр. 2-1403
	SendCmd	Передача команды	стр. 2-1419

Тип	Команда	Имя	Стр.
	NX_SerialSend	Передача данных в беспротокольном режиме	стр. 2-1432
	NX_SerialRcv	Прием данных в беспротокольном режиме	стр. 2-1445
	NX_ModbusRtuCmd	Передача команды общего назначения по протоколу Modbus RTU	стр. 2-1460
	NX_ModbusRtuRead	Передача команды чтения по протоколу Modbus RTU	стр. 2-1472
	NX_ModbusRtuWrite	Передача команды записи по протоколу Modbus RTU	стр. 2-1485
	NX_SerialSigCtl	Включение/выключение сигнала управления последовательного порта	стр. 2-1498
	NX_SerialSigRead	Чтение сигнала управления последовательного порта	стр. 2-1506
	NX_SerialStatusRead	Чтение состояния последовательного порта	стр. 2-1511
	NX_SerialBufClear	Очистка буфера	стр. 2-1516
	NX_SerialStartMon	Запуск мониторинга линии последовательного интерфейса	стр. 2-1527
	NX_SerialStopMon	Остановка мониторинга линии последовательного интерфейса	стр. 2-1532
Команды для работы с картой памяти SD	FileWriteVar	Запись переменной в файл	стр. 2-1538
	FileReadVar	Чтение переменной из файла	стр. 2-1544
	FileOpen	Открытие файла	стр. 2-1550
	FileClose	Закрытие файла	стр. 2-1554
	FileSeek	Поиск в файле	стр. 2-1557
	FileRead	Чтение из файла	стр. 2-1560
	FileWrite	Запись в файл	стр. 2-1569
	FileGets	Получить текстовую строку	стр. 2-1577
	FilePuts	Поместить текстовую строку	стр. 2-1585
	FileCopy	Копирование файла	стр. 2-1594
	FileRemove	Удаление файла	стр. 2-1602
	FileRename	Изменение имени файла	стр. 2-1607
	DirCreate	Создание каталога	стр. 2-1613
	DirRemove	Удаление каталога	стр. 2-1616
	BackupToMemoryCard	Резервное копирование на карту памяти SD	стр. 2-1620
Команды с использованием меток времени	NX_DOutTimeStamp	Запись в дискретный выход с указанием метки времени	стр. 2-1636
	NX_AryDOutTimeStamp	Запись в дискретный выход из массива с указанием метки времени	стр. 2-1643

A-4 Команды, связанные с ошибками обмена сообщениями NX

Если одновременно будет выполняться слишком много указанных ниже команд, произойдет *ошибка обмена сообщениями NX*. При возникновении *ошибки обмена сообщениями NX* необходимо уменьшить количество таких команд, выполняемых одновременно. Условия возникновения *ошибки обмена сообщениями NX* зависят от разных факторов, например от загруженности канала передачи данных.

Классификация	Команда	Имя	Стр.
Команды для управления системой	RestartNXUnit	Перезапуск модуля NX	стр. 2-992
	NX_ChangeWriteMode	Перевод модуля NX в режим записи	стр. 2-998
	NX_SaveParam	Сохранение параметров модуля NX	стр. 2-1004
	NX_ReadTotalPowerOnTime	Чтение общего времени работы модуля NX	стр. 2-1013
Команды для обмена данными по интерфейсу EtherCAT	EC_CoESDOWrite	Запись в SDO CoE в EtherCAT	стр. 2-1060
	EC_CoESDORead	Чтение из SDO CoE в EtherCAT	стр. 2-1064
	EC_StartMon	Запуск мониторинга пакетов EtherCAT	стр. 2-1070
	EC_StopMon	Остановка мониторинга пакетов EtherCAT	стр. 2-1076
	EC_SaveMon	Сохранение пакетов EtherCAT	стр. 2-1078
	EC_CopyMon	Передача пакетов EtherCAT	стр. 2-1080
	EC_DisconnectSlave	Отсоединение ведомого устройства EtherCAT	стр. 2-1082
	EC_ConnectSlave	Подсоединение ведомого устройства EtherCAT	стр. 2-1091
	EC_ChangeEnableSetting	Активация/деактивация ведомого устройства EtherCAT	стр. 2-1093
	NX_WriteObj	Запись в объект модуля NX	стр. 2-1114
	NX_ReadObj	Чтение объекта модуля NX	стр. 2-1131
Команды для обмена данными по интерфейсу IO-Link	IOL_ReadObj	Чтение из объекта устройства IO-Link	стр. 2-1142
	IOL_WriteObj	Запись в объект устройства IO-Link	стр. 2-1151

Информация о версии

Для возникновения ошибки обмена сообщениями NX требуется модуль ЦПУ с версией модуля 1.05 или более поздней и Sysmac Studio версии 1.06 или выше.

A-5 Коды прерывания SDO

В следующей таблице в качестве справочной информации приводятся коды прерывания SDO для интерфейса связи EtherCAT. Коды прерывания, которые используются при реальном обмене данными, указываются ведомыми устройствами. При программировании обмена данными см. информацию в руководствах по соответствующим ведомым устройствам.

Значение	Смысл
16#05030000	Состояние переключающего бита не изменилось
16#05040000	Превышение времени ожидания протокола SDO
16#05040001	Недопустимый или неизвестный спецификатор команды клиента/сервера
16#05040005	Недостаточно памяти
16#06010000	Доступ к объекту не поддерживается
16#06010001	Попытка чтения из объекта, доступного только для записи
16#06010002	Попытка записи в объект, доступный только для чтения
16#06020000	Объект не существует в каталоге объектов
16#06040041	Объект не может быть сопоставлен в PDO
16#06040042	Количество и длины сопоставляемых объектов приведут к превышению длины PDO
16#06040043	Общая несовместимость параметров
16#06040047	Общая внутренняя несовместимость в устройстве
16#06060000	Сбой доступа из-за аппаратной ошибки
16#06070010	Тип данных не соответствует, длина сервисного параметра не соответствует
16#06070012	Тип данных не соответствует, длина сервисного параметра слишком велика
16#06070013	Тип данных не соответствует, длина сервисного параметра слишком мала
16#06090011	Подындекс не существует
16#06090030	Диапазон значений параметра превышен (только для доступа с целью записи)
16#06090031	Записанное значение параметра слишком велико
16#06090032	Записанное значение параметра слишком мало
16#06090036	Максимальное значение меньше минимального значения
16#08000000	Общая ошибка
16#08000020	Данные не могут быть переданы или сохранены в приложение
16#08000021	Данные не могут быть переданы или сохранены в приложение из-за локального управления*1
16#08000022	Данные не могут быть переданы или сохранены в приложении из-за текущего состояния устройства
16#08000023	Сбой динамического создания словаря объектов, или словарь объектов отсутствует

*1. Это внутреннее состояние, уникальное для конкретного ведомого устройства.

Источник: Спецификации стандарта EtherCAT. Часть 6. Спецификации протоколов прикладного уровня.
Номер документа: ETG.1000.6 S (R) V1.0.2

A-6 Сведения о версиях

В данном приложении приводится список команд с изменившимися характеристиками, а также команд, которые были добавлены в разных версиях модулей ЦПУ и в разных версиях Sysmac Studio.
В нем также описываются действия, которые нужно предпринять при отображении указанного ниже сообщения об ошибке в Sysmac Studio версии 1.02.

- Команда может вызвать непредусмотренные действия.

A-6-1 Команды с изменившимися характеристиками и новые команды

Поддерживаемые команды и их характеристики зависят от версии модуля ЦПУ и версии ПО Sysmac Studio. Эти команды перечислены в таблице ниже.
Если для команды указана версия модуля ЦПУ и версия Sysmac Studio, значит для команды требуются обе версии.

Тип	Команда	Имя	Новая/ изменившаяся	Версии		Стр.
				Модуль ЦПУ	Sysmac Studio	
Команды для инструкций языка ST	FOR	Начало цикла	Изменена	---	Вер. 1.08	стр. 2-48
Входные битовые команды	R_TRIG	Триггер по полож. фронту	Изменена	Вер. 1.02	---	стр. 2-50
Выходные битовые команды	RS	Удержание с приоритетом сброса	Изменена	---	Вер. 1.03	стр. 2-58
	SR	Удержание с приоритетом установки	Изменена	---	Вер. 1.03	стр. 2-61
Команды сравнения	EQ (=)	Равно	Изменена	---	Вер. 1.02	стр. 2-108
	NE (<>)	Не равно	Изменена	---	Вер. 1.02	стр. 2-111
	LT (<)	Меньше	Изменена	---	Вер. 1.02	стр. 2-114
	LE (<=)	Меньше или равно	Изменена	---	Вер. 1.02	стр. 2-114
	GT (>)	Больше	Изменена	---	Вер. 1.02	стр. 2-114
	GE (>=)	Больше или равно	Изменена	---	Вер. 1.02	стр. 2-114
	ZoneCmp	Попадание в диапазон	Изменена	Вер. 1.01	Вер. 1.02	стр. 2-128
Команды счетчиков	CTD	Обратный счетчик	Изменена	---	Вер. 1.03	стр. 2-170
	CTD_**	Групповой обратный счетчик	Изменена	---	Вер. 1.03	стр. 2-173
	CTU	Прямой счетчик	Изменена	---	Вер. 1.03	стр. 2-176
	CTU_**	Групповой прямой счетчик	Изменена	---	Вер. 1.03	стр. 2-179

Тип	Команда	Имя	Новая/ изме-нив-шаяся	Версии		Стр.
				Модуль ЦПУ	Sysmac Studio	
	CTUD	Прямой-обратный счетчик	Измене-на	---	Вер. 1.03	стр. 2-182
	CTUD_**	Групповой прямой-обратный счетчик	Измене-на	---	Вер. 1.03	стр. 2-187
Команды математических операций	EXPT(**)	Возведение в степень	Измене-на	Вер. 1.16	Вер. 1.20	стр. 2-247
Команды преобразования типов данных	EnumToNum	Перечисление в целое число	Новая	Вер. 1.02	Вер. 1.03	стр. 2-361
	NumToEnum	Целое число в перечисление	Новая	Вер. 1.02	Вер. 1.03	стр. 2-363
Команды выбора	SEL	Двоичный выбор	Измене-на	Вер. 1.02	Вер. 1.03	стр. 2-382
	MUX	Мультиплексор	Измене-на	Вер. 1.02	Вер. 1.03	стр. 2-385
	AryMax	Максимум массива	Измене-на	Вер. 1.01	Вер. 1.02	стр. 2-399
	AryMin	Минимум массива	Измене-на	Вер. 1.01	Вер. 1.02	стр. 2-399
	ArySearch	Поиск по массиву	Измене-на	Вер. 1.01	Вер. 1.02	стр. 2-402
Команды сдвига	SHL	Сдвиг влево на N битов	Измене-на	Вер. 1.02	Вер. 1.03	стр. 2-451
	SHR	Сдвиг вправо на N битов	Измене-на	Вер. 1.02	Вер. 1.03	стр. 2-451
	ROL	Циклический сдвиг на N битов влево	Измене-на	Вер. 1.02	Вер. 1.03	стр. 2-457
	ROR	Циклический сдвиг на N битов вправо	Измене-на	Вер. 1.02	Вер. 1.03	стр. 2-457
Команды преобразования	UTF8ToSJIS	Преобразование кодировки UTF-8 в SJIS	Новая	Вер. 1.01	Вер. 1.02	стр. 2-484
	SJISToUTF8	Преобразование кодировки SJIS в UTF-8	Новая	Вер. 1.01	Вер. 1.02	стр. 2-486
	PWLApproxNoLineChk	Аппроксимация ломаной линии без проверки данных ломаной линии	Новая	Вер. 1.03	Вер. 1.04	стр. 2-488
	PWLLineChk	Проверка данных ломаной линии	Новая	Вер. 1.03	Вер. 1.04	стр. 2-494
	PackWord	Объединение двух байтов	Новая	Вер. 1.12	Вер. 1.16	стр. 2-556
	PackDword	Объединение четырех байтов	Новая	Вер. 1.12	Вер. 1.16	стр. 2-558
	LOWER_BOUND	Получить минимальное значение индекса массива	Новая	Версия 1.18	Версия 1.22	стр. 2-560
	UPPER_BOUND	Получить максимальное значение индекса массива	Новая	Версия 1.18	Версия 1.22	стр. 2-560

Тип	Команда	Имя	Новая/ изменившаяся	Версии		Стр.
				Модуль ЦПУ	Sysmac Studio	
Команды для работы со стеком и таблицами	RecSearch	Поиск записи	Изменена	Вер. 1.01	Вер. 1.02	стр. 2-584
	RecRangeSearch	Поиск диапазона записей	Изменена	Вер. 1.01	Вер. 1.02	стр. 2-589
	RecSort	Сортировка записей	Изменена	Вер. 1.01	Вер. 1.02	стр. 2-594
	RecNum	Получить количество записей	Изменена	Вер. 1.01	Вер. 1.02	стр. 2-601
				Вер. 1.02	Вер. 1.03	
	RecMax	Поиск записи с максимальным значением	Изменена	Вер. 1.01	Вер. 1.02	стр. 2-604
	RecMin	Поиск записи с минимальным значением	Изменена	Вер. 1.01	Вер. 1.02	стр. 2-604
Команды для обработки текстовых строк	AddDelimiter	Поместить в текстовую строку с разделителями	Новая	Вер. 1.02	Вер. 1.03	стр. 2-651
	SubDelimiter	Получить текстовые строки без разделителей	Новая	Вер. 1.02	Вер. 1.03	стр. 2-664
Команды для обработки времени и времени суток	TruncTime	Усечение времени	Новая	Вер. 1.01	Вер. 1.02	стр. 2-748
	TruncDt	Усечение даты и времени	Новая	Вер. 1.01	Вер. 1.02	стр. 2-753
	TruncTod	Усечение времени суток	Новая	Вер. 1.01	Вер. 1.02	стр. 2-757
Команды для аналогового регулирования	PIDAT_HeatCool	ПИД-регулятор нагрева/охлаждения с автонастройкой	Новая	Вер. 1.08	Вер. 1.09	стр. 2-796
	TimeProportionalOut	Выход широтно-импульсного регулирования	Новая	Вер. 1.02	Вер. 1.03	стр. 2-839
	LimitAlarm_**	Группа сигнализации аварии выхода за верхний/нижний предел	Новая	Вер. 1.02	Вер. 1.03	стр. 2-860
	LimitAlarmDv_**	Группа сигнализации аварии верхнего/ нижнего отклонения	Новая	Вер. 1.02	Вер. 1.03	стр. 2-865
	LimitAlarmDvStbySeq_**	Группа сигнализации аварии верхнего/ нижнего отклонения с контролем последовательности	Новая	Вер. 1.02	---	стр. 2-871
	ScaleTrans	Изменение масштаба	Новая	Вер. 1.05	Вер. 1.06	стр. 2-891
	AC_StepProgram	Поэтапная программа	Новая	Вер. 1.06	Вер. 1.07	стр. 2-894
			Изменена	Вер. 1.21/ вер.1.32	Вер.1.28	

Тип	Команда	Имя	Новая/ изме-нив-шаяся	Версии		Стр.
				Модуль ЦПУ	Sysmac Studio	
Команды для управления системой	ResetMCError	Сброс ошибки в Motion Control	Измене-на	Вер. 1.02	Вер. 1.03	стр. 2-957
				Вер. 1.10	Вер. 1.12	
	GetECError	Получить состояние ошибки в EtherCAT	Измене-на	Вер. 1.02	Вер. 1.03	стр. 2-968
	ResetNXBError	Сброс ошибки в NX Bus	Новая	Вер. 1.13	Вер. 1.17	стр. 2-971
	GetNXBError	Получить состояние ошибки в NX Bus	Новая	Вер. 1.13	Вер. 1.17	стр. 2-973
	GetNXUnitError	Получить состояние ошибки в модуле NX	Новая	Вер. 1.13	Вер. 1.17	стр. 2-975
	RestartNXUnit	Перезапуск модуля NX	Новая	Вер. 1.05	Вер. 1.06	стр. 2-992
			Измене-на	Вер. 1.07	Вер. 1.08	
	NX_ChangeWrite Mode	Перевод модуля NX в режим записи	Новая	Вер. 1.05	Вер. 1.06	стр. 2-998
	NX_SaveParam	Сохранение парамет-ров модуля NX	Новая	Вер. 1.05	Вер. 1.06	стр. 2-1004
	PLC_ReadTotalPo werOnTime	Чтение общего време-ни работы ПЛК	Новая	Вер. 1.13	Вер. 1.17	стр. 2-1010
	NX_ReadTotalPow erOnTime	Чтение общего време-ни работы модуля NX	Новая	Вер. 1.10	Вер. 1.12	стр. 2-1013
Команды для управления программами	PrgStart	Активация программы	Новая	Вер. 1.08	Вер. 1.09	стр. 2-1022
	PrgStop	Деактивация програм-мы	Новая	Вер. 1.08	Вер. 1.09	стр. 2-1032
	PrgStatus	Чтение статуса про-граммы	Новая	Вер. 1.08	Вер. 1.09	стр. 2-1052
Команды для обмена дан-ными по интерфейсу EtherCAT	EC_StartMon	Запуск мониторинга пакетов EtherCAT	Измене-на	Вер. 1.10	Вер 1.12*1	стр. 2-1070
			Измене-на	Вер 1.40 *2	Вер. 1.29	
	EC_StopMon	Остановка мониторин-га пакетов EtherCAT	Измене-на	Вер. 1.10	Вер 1.12*1	стр. 2-1076
			Измене-на	Вер 1.40 *2	Вер. 1.29	
	EC_SaveMon	Сохранение пакетов EtherCAT	Измене-на	Вер. 1.10	Вер 1.12*1	стр. 2-1078
			Измене-на	Вер 1.40 *2	Вер. 1.29	
	EC_CopyMon	Передача пакетов EtherCAT	Измене-на	Вер. 1.10	Вер 1.12*1	стр. 2-1080
			Измене-на	Вер 1.40 *2	Вер. 1.29	
	EC_ChangeEnabl eSetting	Активация/деактива-ция ведомого устройства EtherCAT	Новая	Вер. 1.04	Вер. 1.05	стр. 2-1093
	NX_WriteObj	Запись в объект модуля NX	Новая	Вер. 1.05	Вер. 1.06	стр. 2-1114
	NX_ReadObj	Чтение объекта модуля NX	Новая	Вер. 1.05	Вер. 1.06	стр. 2-1131

Тип	Команда	Имя	Новая/ изме-нив-шаяся	Версии		Стр.
				Модуль ЦПУ	Sysmac Studio	
Команды для обмена данными по интерфейсу IO-Link	IOL_ReadObj	Чтение из объекта устройства IO-Link	Новая	Вер. 1.12	Вер. 1.16	стр. 2-1142
	IOL_WriteObj	Запись в объект устройства IO-Link	Новая	Вер. 1.12	Вер. 1.16	стр. 2-1151
Команды для обмена данными по интерфейсу EtherNet/IP	CIPOpenWithDataSize	Открытие CIP-соединения класса 3 с указанным объемом данных	Новая	Вер. 1.06	Вер. 1.07	стр. 2-1174
	CIPSend	Передача явного сообщения класса 3	Изменена	Вер. 1.11	Вер. 1.15	стр. 2-1191
	CIPUCMMSend	Передача явного сообщения UCMM	Изменена	Вер. 1.11	Вер. 1.15	стр. 2-1214
	SktUDPCreate	Создание сокета UDP	Изменена	Вер. 1.03	---	стр. 2-1227
			Изменена	Вер. 1.10	Вер. 1.13	
	SktTCPAccept	Принятие сокета TCP	Изменена	Вер. 1.03	---	стр. 2-1243
	SktTCPConnect	Соединение с сокетом TCP	Изменена	Вер. 1.03	---	стр. 2-1246
	SktSetOption	Настройка дополнительного параметра сокета TCP	Новая	Вер. 1.12*3	Вер. 1.16*3	стр. 2-1271
	ModbusTCPCmd	Передача команды общего назначения по протоколу Modbus TCP	Новая	Вер. 1.30	Вер. 1.23	стр. 2-1277
	ModbusTCPRead	Передача команды чтения по протоколу Modbus TCP	Новая	Вер. 1.30	Вер. 1.23	стр. 2-1286
	ModbusTCPWrite	Передача команды записи по протоколу Modbus TCP	Новая	Вер. 1.30	Вер. 1.23	стр. 2-1295
	ChangeIPAdr	Изменение IP-адреса	Новая	Вер. 1.02	Вер. 1.03	стр. 2-1303
			Изменена	Вер. 1.10	Вер. 1.13	
	ChangeFTPAccount	Изменение учетной записи FTP	Новая	Вер. 1.02	Вер. 1.03	стр. 2-1312
			Изменена	Вер. 1.10	Вер. 1.13	
	ChangeNTPServerAdr	Изменение адреса сервера NTP	Новая	Вер. 1.02	Вер. 1.03	стр. 2-1316
			Изменена	Вер. 1.10	Вер. 1.13	
	FTPGetFileList	Получение списка файлов на сервере FTP	Новая	Вер. 1.08	Вер. 1.09	стр. 2-1321
			Изменена	Вер. 1.09	Вер. 1.10	
			Изменена	Вер. 1.16	---	

Тип	Команда	Имя	Новая/ изме-нив-шаяся	Версии		Стр.
				Модуль ЦПУ	Sysmac Studio	
	FTPGetFile	Получение файла с сервера FTP	Новая	Вер. 1.08	Вер. 1.09	стр. 2-1337
			Измене-на	Вер. 1.09	Вер. 1.10	
			Измене-на	Вер. 1.16	---	
	FTPPutFile	Размещение файла на сервере FTP	Новая	Вер. 1.08	Вер. 1.09	стр. 2-1347
			Измене-на	Вер. 1.09	Вер. 1.10	
			Измене-на	Вер. 1.16	---	
	FTPRemoveFile	Удаление файла с сервера FTP	Новая	Вер. 1.08	Вер. 1.09	стр. 2-1358
			Измене-на	Вер. 1.09	Вер. 1.10	
			Измене-на	Вер. 1.16	---	
	FTPRemoveDir	Удаление каталога с сервера FTP	Новая	Вер. 1.08	Вер. 1.09	стр. 2-1369
			Измене-на	Вер. 1.09	Вер. 1.10	
			Измене-на	Вер. 1.16	---	
Команды для обмена данными по последовательному интерфейсу	SerialRcvNoClear *4	Прием через последовательный порт SCU без очистки буфера приема	Новая	Вер. 1.03	Вер. 1.04	стр. 2-1403
	NX_SerialSend	Передача данных в беспротокольном режиме	Новая	Вер. 1.11	Вер. 1.15	стр. 2-1432
	NX_SerialRcv	Прием данных в беспротокольном режиме	Новая	Вер. 1.11	Вер. 1.15	стр. 2-1445
	NX_ModbusRtuCmd	Передача команды общего назначения по протоколу Modbus RTU	Новая	Вер. 1.11	Вер. 1.15	стр. 2-1460
	NX_ModbusRtuRead	Передача команды чтения по протоколу Modbus RTU	Новая	Вер. 1.11	Вер. 1.15	стр. 2-1472
	NX_ModbusRtuWrite	Передача команды записи по протоколу Modbus RTU	Новая	Вер. 1.11	Вер. 1.15	стр. 2-1485
	NX_SerialSigCtl	Включение/выключение сигнала управления последовательного порта	Новая	Вер. 1.11	Вер. 1.15	стр. 2-1498
	NX_SerialSigRead	Чтение сигнала управления последовательного порта	Новая	Вер. 1.13	Вер. 1.17	стр. 2-1506
	NX_SerialStatusRead	Чтение состояния последовательного порта	Новая	Вер. 1.13	Вер. 1.17	стр. 2-1511
	NX_SerialBufClear	Очистка буфера	Новая	Вер. 1.11	Вер. 1.15	стр. 2-1516
	NX_SerialStartMon	Запуск мониторинга линии последовательного интерфейса	Новая	Вер. 1.11	Вер. 1.15	стр. 2-1527

Тип	Команда	Имя	Новая/ изме-нив-шаяся	Версии		Стр.
				Модуль ЦПУ	Sysmac Studio	
	NX_SerialStopMon	Остановка мониторинга линии последовательного интерфейса	Новая	Вер. 1.11	Вер. 1.15	стр. 2-1532
Команды для работы с картой памяти SD	BackupToMemoryCard	Резервное копирование на карту памяти SD	Новая	Вер. 1.08	Вер. 1.09	стр. 2-1620
Команды с использованием меток времени	NX_DOutTimeStamp	Запись в дискретный выход с указанием метки времени	Новая	Вер. 1.06	Вер. 1.07	стр. 2-1636
	NX_AryDOutTimeStamp	Запись в дискретный выход из массива с указанием метки времени	Новая	Вер. 1.06	Вер. 1.07	стр. 2-1643
Прочие команды	GetMyTaskInterval	Чтение периода текущей задачи	Новая	Вер. 1.08	Вер. 1.09	стр. 2-1664
	ActEventTask	Инициация событийной задачи	Новая	Вер. 1.03	Вер. 1.04	стр. 2-1675

*1. Для модуля ЦПУ NJ101 требуется ПО Sysmac Studio версии 1.13 или выше.
*2. Эту команду нельзя использовать, если в проекте указана версия модуля 1.40 или более поздняя.
*3. В случае модуля ЦПУ NX1P2 для использования этой команды требуются модуль ЦПУ с версией модуля 1.14 или более поздней и Sysmac Studio версии 1.18 или выше.
*4. Для использования команды SerialRcvNoClear требуются модуль ЦПУ с версией модуля не ниже 1.03, Sysmac Studio версии 1.04 или выше и модуль последовательного интерфейса с версией модуля не ниже 2.1.

A-6-2 Действия при отображении сообщения об ошибке «Команда может вызвать непредусмотренные действия»

В Sysmac Studio может отображаться следующее сообщение об ошибке:

- Команда может вызвать непредусмотренные действия. Дополнительную информацию см. в руководстве по командам программирования.

Данное сообщение возникает из-за ограничений в программе пользователя. Для его устранения могут потребоваться изменения в программе пользователя.
В этом приложении описываются условия, при которых может отображаться данное сообщение об ошибке, и требуемые изменения в программе пользователя.

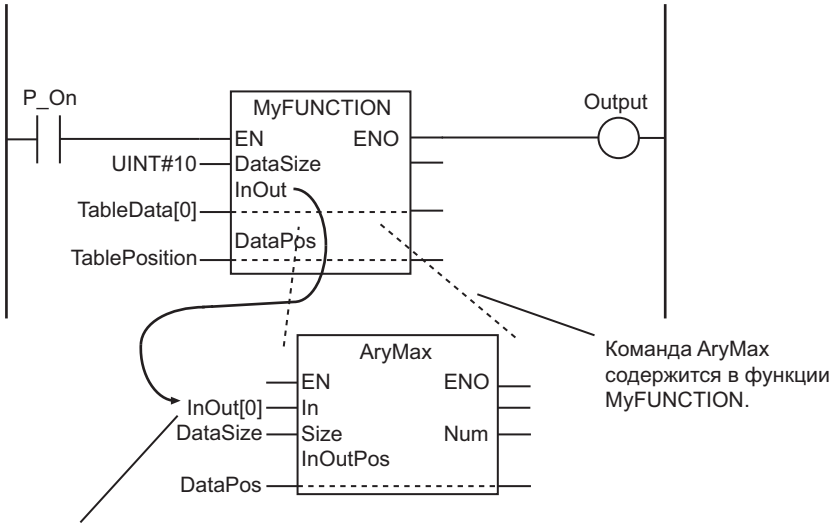
Информация о версии

Это сообщение об ошибке отображается только для Sysmac Studio версии 1.02.

Условия отображения сообщения об ошибке

Данное сообщение об ошибке отображается, когда входная-выходная переменная в функции или функциональном блоке передается определенной переменной определенной команды в той же функции или функциональном блоке.
Список команд и переменных, для которых отображается сообщение об ошибке, приводится в таблице ниже.

Пример программы, для которой отображается сообщение об ошибке:



В этом примере входная-выходная переменная ***InOut*** функции MyFUNCTION передается в переменную ***In*** команды AryMax.

Команды и переменные, для которых отображается сообщение об ошибке

Это сообщение об ошибке отображается только для определенных переменных в определенных командах. Такие команды и переменные перечислены в следующей таблице.

Тип	Команда	Имя	Функция	Стр.
Команды сравнения	TableCmp	Таблица сравнения	Table и AryOut	стр. 2-131
	AryCmpEQ	Сравнение массивов: равно	In1, In2 и AryOut	стр. 2-134
	AryCmpNE	Сравнение массивов: не равно	In1, In2 и AryOut	стр. 2-134
	AryCmpLT	Сравнение массивов: меньше	In1, In2 и AryOut	стр. 2-137
	AryCmpLE	Сравнение массивов: меньше или равно	In1, In2 и AryOut	стр. 2-137
	AryCmpGT	Сравнение массивов: больше	In1, In2 и AryOut	стр. 2-137
	AryCmpGE	Сравнение массивов: больше или равно	In1, In2 и AryOut	стр. 2-137
	AryCmpEQV	Сравнение массива со значением: равно	In1 и AryOut	стр. 2-140
	AryCmpNEV	Сравнение массива со значением: не равно	In1 и AryOut	стр. 2-140
	AryCmpLTV	Сравнение массива со значением: меньше	In1 и AryOut	стр. 2-143
	AryCmpLEV	Сравнение массива со значением: меньше или равно	In1 и AryOut	стр. 2-143
	AryCmpGTV	Сравнение массива со значением: больше	In1 и AryOut	стр. 2-143
	AryCmpGEV	Сравнение массива со значением: больше или равно	In1 и AryOut	стр. 2-143

Тип	Команда	Имя	Функция	Стр.
Команды математических операций	AryAdd	Сложение массивов	In1, In2 и AryOut	стр. 2-257
	AryAddV	Прибавление значения к массиву	In1 и AryOut	стр. 2-259
	ArySub	Вычитание массивов	In1, In2 и AryOut	стр. 2-261
	ArySubV	Вычитание значения из массива	In1 и AryOut	стр. 2-263
	AryMean	Среднее значение массива	In	стр. 2-265
	ArySD	Среднеквадратическое отклонение элементов массива	In	стр. 2-267
Команды преобразования двоично-десятичных значений	AryToBCD	Преобразование массива в BCD	In1 и AryOut	стр. 2-293
	AryToBin	Преобразование массива в беззнаковые целые	In1 и AryOut	стр. 2-295
Команды обработки битовых строк	AryAnd	Логическое И массивов	In1, In2 и AryOut	стр. 2-377
	AryOr	Логическое ИЛИ массивов	In1, In2 и AryOut	стр. 2-377
	AryXor	Логическое исключающее ИЛИ массивов	In1, In2 и AryOut	стр. 2-377
	AryXorN	Логическое исключающее ИЛИ-НЕ массивов	In1, In2 и AryOut	стр. 2-377
Команды выбора	AryMax	Максимум массива	In	стр. 2-399
	AryMin	Минимум массива	In	стр. 2-399
	ArySearch	Поиск по массиву	In	стр. 2-402
Команды перемещения данных	TransBits	Перемещение битов	InOut	стр. 2-413
	AryExchange	Обмен данными массивов	InOut1 и InOut2	стр. 2-422
	AryMove	Перемещение массива	In1 и AryOut	стр. 2-424
	Clear	Инициализация	InOut	стр. 2-426
Команды сдвига	AryShiftReg	Регистр сдвига	InOut	стр. 2-442
	AryShiftRegLR	Реверсивный регистр сдвига	InOut	стр. 2-445
	ArySHL	Сдвиг массива влево на N элементов	InOut	стр. 2-448
	ArySHR	Сдвиг массива вправо на N элементов	InOut	стр. 2-448
	NSHLC	Сдвиг на N битов влево с переносом	InOut	стр. 2-454
	NSHRC	Сдвиг на N битов вправо с переносом	InOut	стр. 2-454
Команды преобразования	Decoder	Битовый дешифратор	InOut	стр. 2-467
	Encoder	Битовый шифратор	In	стр. 2-470
	ColmToLine_**	Группа преобразования столбца в строку	In	стр. 2-474

Тип	Команда	Имя	Функция	Стр.
	LineToColm	Преобразование строки в столбец	InOut	стр. 2-476
	PWLApprox	Аппроксимация ломаной линии	Line	стр. 2-488
	MovingAverage	Скользящее среднее	Buf	стр. 2-497
	StringToAry	Преобразование текстовой строки в массив	AryOut	стр. 2-529
	AryToString	Преобразование массива в текстовую строку	In	стр. 2-531
	DispartDigit	Разделение на 4-битные блоки	AryOut	стр. 2-533
	UniteDigit_**	Группа объединения 4-битных блоков	In	стр. 2-535
	Dispart8Bit	Разделение данных на байты	AryOut	стр. 2-537
	Unite8Bit_**	Группа объединения байтов данных	In	стр. 2-539
	ToAryByte	Преобразование в байтовый массив	In1 и AryOut	стр. 2-541
	AryByteTo	Преобразование из байтового массива	In и OutVal	стр. 2-547
	SizeOfAry	Получить количество элементов массива	In	стр. 2-554
Команды для работы со стеком и таблицами	StackPush	Ввод в стек	InOut	стр. 2-566
	StackFIFO	Первым вошел — первым вышел	InOut и OutVal	стр. 2-575
	StackLIFO	Последним вошел — первым вышел	InOut и OutVal	стр. 2-575
	StackIns	Вставка в стек	InOut	стр. 2-579
	StackDel	Удаление из стека	InOut	стр. 2-582
	RecSearch	Поиск записи	In, Member и InOutPos	стр. 2-584
	RecRangeSearch	Поиск диапазона записей	In, Member и InOutPos	стр. 2-589
	RecSort	Сортировка записей	InOut и Member	стр. 2-594
	RecNum	Получить количество записей	In и Member	стр. 2-601
	RecMax	Поиск записи с максимальным значением	In, Member и InOutPos	стр. 2-604
	RecMin	Поиск записи с минимальным значением	In, Member и InOutPos	стр. 2-604
Команды вычисления контрольной суммы	AryLRC_**	Группа расчета LRC массива	In	стр. 2-618
	AryCRCCCITT	Расчет CRC-CCITT массива	In	стр. 2-620
	AryCRC16	Расчет CRC-16 массива	In	стр. 2-622
Команды для управления системой	SetAlarm	Создание ошибки пользователя	Info1 и Info2	стр. 2-936
	SetInfo	Создание информации пользователя	Info1 и Info2	стр. 2-983

Тип	Команда	Имя	Функция	Стр.
Команды для обмена данными по интерфейсу EtherCAT	EC_CoESDOWrite	Запись в SDO CoE в EtherCAT	WriteDat	стр. 2-1060
	EC_CoESDORead	Чтение из SDO CoE в EtherCAT	ReadDat	стр. 2-1064
Команды для обмена данными по интерфейсу EtherNet/IP	CIPRead	Чтение переменной посредством явного сообщения класса 3	DstDat	стр. 2-1178
	CIPWrite	Запись переменной посредством явного сообщения класса 3	SrcDat	стр. 2-1184
	CIPSend	Передача явного сообщения класса 3	ServiceDat и RespServiceDat	стр. 2-1191
	CIPUCMMRead	Чтение переменной посредством явного сообщения UCMM	DstDat	стр. 2-1200
	CIPUCMMWrite	Запись переменной посредством явного сообщения UCMM	SrcDat	стр. 2-1206
	CIPUCMMSend	Передача явного сообщения UCMM	ServiceDat и RespServiceDat	стр. 2-1214
	SktUDPRcv	Прием через сокет UDP	RcvDat	стр. 2-1236
	SktUDPSend	Передача через сокет UDP	SendDat	стр. 2-1240
	SktTCPRcv	Прием через сокет TCP	RcvDat	стр. 2-1255
	SktTCPSend	Передача через сокет TCP	SendDat	стр. 2-1259
Команды для обмена данными по последовательному интерфейсу	ExecPMCR	Макрос протокола	SrcDat и DstDat	стр. 2-1376
	SerialSend	Передача через последовательный порт SCU	SrcDat	стр. 2-1391
	SerialRcv	Прием через последовательный порт SCU	DstDat	стр. 2-1403
	SendCmd	Передача команды	CmdDat и RespDat	стр. 2-1419
Команды для работы с картой памяти SD	FileWriteVar	Запись переменной в файл	WriteVar	стр. 2-1538
	FileReadVar	Чтение переменной из файла	ReadVar	стр. 2-1544
	FileRead	Чтение из файла	ReadBuf	стр. 2-1560
	FileWrite	Запись в файл	WriteBuf	стр. 2-1569

Тип	Команда	Имя	Функция	Стр.
Команды управления движением	MC_SetCamTableProperty	Настройка параметров таблицы кулачка	CamTable	*1
	MC_SaveCamTable	Сохранение таблицы кулачка	CamTable	*1
	MC_Write	Запись настройки управления движением	Target и SettingValue	*1
	MC_CamIn	Запуск кулачковой передачи	CamTable	*1
	MC_ChangeAxesInGroup	Изменение осей в группе	Axes	*1
Прочие команды	ChkRange	Проверка на принадлежность заданному диапазону	Val	стр. 2-1658

*1. Дополнительные сведения см. в: *Серия NJ/NX — Команды программирования для управления движением. Справочное руководство (Cat. No. W508)*.

Исправление программы пользователя

Чтобы указанное сообщение об ошибке не отображалось, в программу пользователя необходимо внести соответствующие исправления.

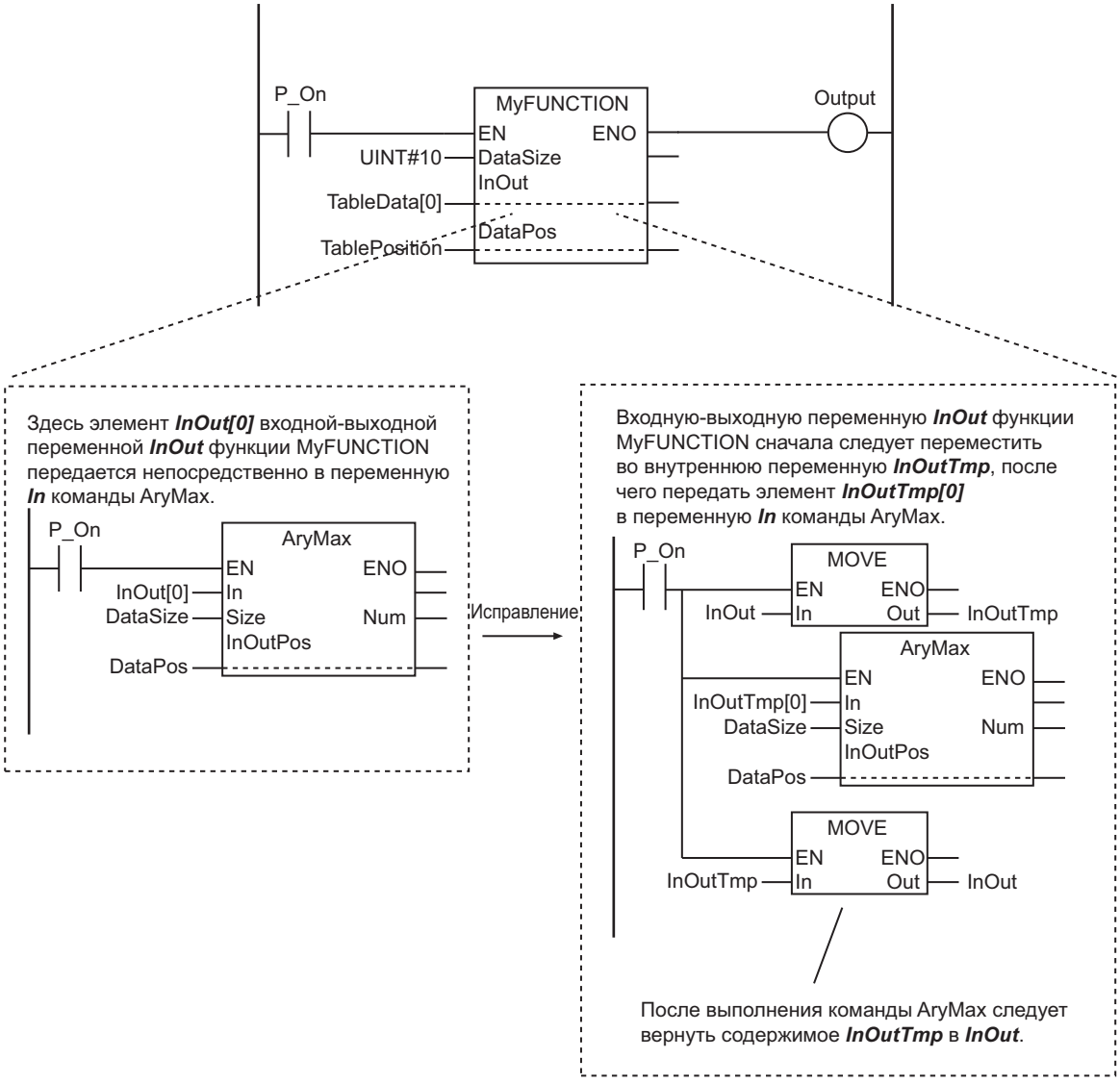
Программу пользователя можно исправить одним из двух указанных ниже способов.

- Скопировать входную-выходную переменную во внутреннюю переменную в функции или функциональном блоке, а затем передать внутреннюю переменную в команду.
- Разместить команду за пределами функции или функционального блока.

● Передача входной-выходной переменной во внутреннюю переменную

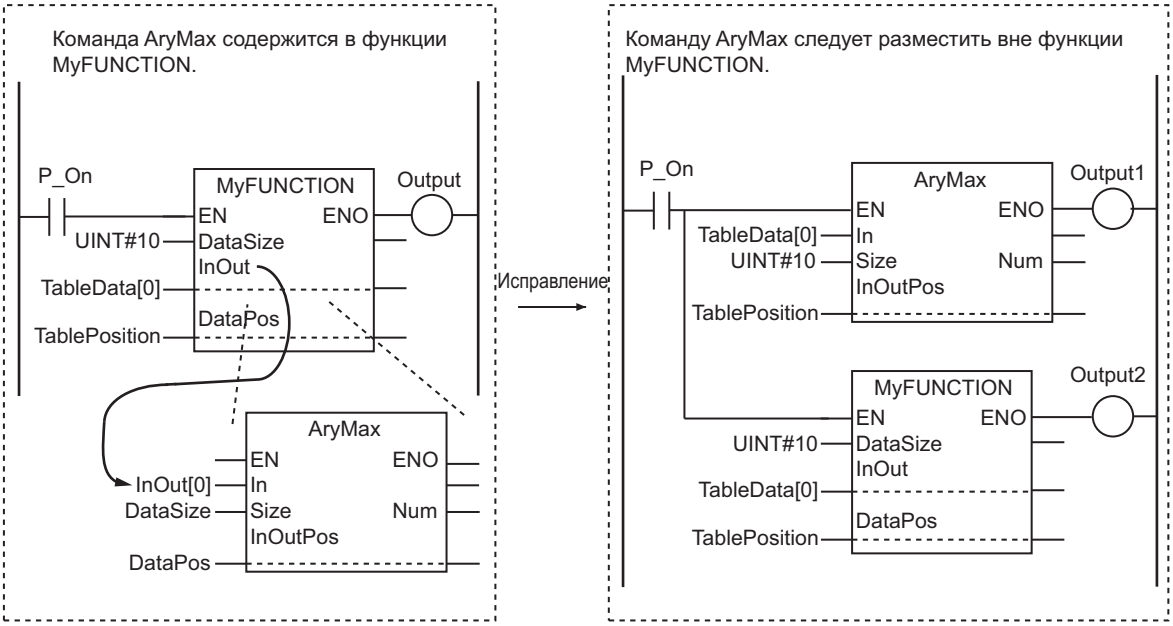
Входную-выходную переменную функции или функционального блока не следует передавать напрямую в команду. Ее сначала нужно скопировать во внутреннюю переменную.
Если значение внутренней переменной изменится во время выполнения команды, оно будет скопировано из внутренней переменной во входную-выходную переменную после выполнения команды.

Этот метод, однако, нельзя использовать для команды Clear. Если нужно использовать команду Clear, ее следует разместить вне функции или функционального блока.



● Размещение команды за пределами функции или функционального блока

Проблему можно устранить, разместив команду не внутри, а за пределами функции или функционального блока.



Когда сообщение об ошибке может быть проигнорировано

В некоторых ситуациях команду можно использовать без проблем, даже если отображается сообщение об ошибке. Можно или нет применять команду, зависит от параметра, который передается во входную-выходную переменную функции или функционального блока.
Эти условия перечислены в следующей таблице.

Применимость команды	Параметр, передаваемый во входную-выходную переменную функции или функционального блока
Можно использовать.	Базовый тип данных, перечисление, массив, структура или объединение
Нельзя использовать.	Один элемент массива либо один член структуры или объединения

● Пример ситуации, в которой команду можно использовать

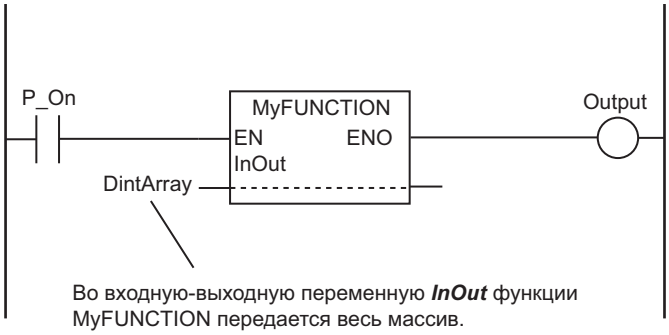
В данном примере во входную-выходную переменную функции или функционального блока передается массив, который затем используется в команде внутри функции или функционального блока.

Переменная, вызываемая функцией MyFUNCTION

Переменная	Тип данных
DintArray	ARRAY[0..9] OF DINT

Переменная в функции MyFUNCTION

Переменная	Тип данных
InOut	ARRAY[0..9] OF DINT



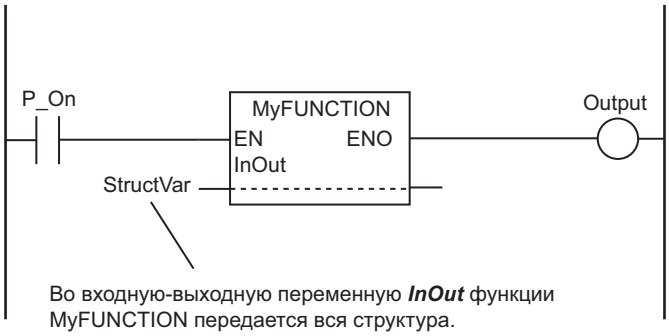
В данном примере во входную-выходную переменную функции или функционального блока передается структура, которая затем используется в команде внутри функции или функционального блока.

Переменная, вызываемая функцией MyFUNCTION

Переменная	Тип данных
StructVar	STRUCT

Переменная в функции MyFUNCTION

Переменная	Тип данных
InOut	STRUCT



● Пример ситуации, в которой команду использовать нельзя

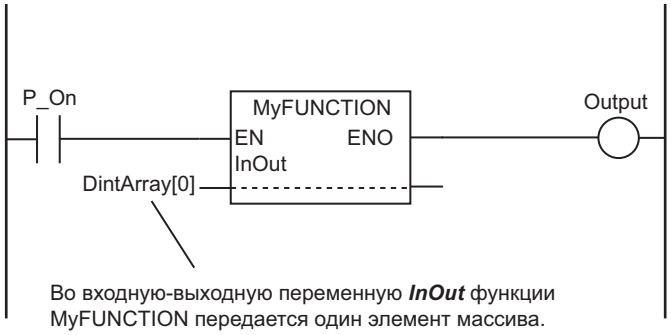
В следующем примере во входную-выходную переменную функции или функционального блока передается один элемент массива, поэтому команду нельзя использовать внутри функции или функционального блока.

Переменная, вызываемая функцией MyFUNCTION

Переменная	Тип данных
DintArray	ARRAY[0..9] OF DINT

Переменная в функции MyFUNCTION

Переменная	Тип данных
InOut	DINT



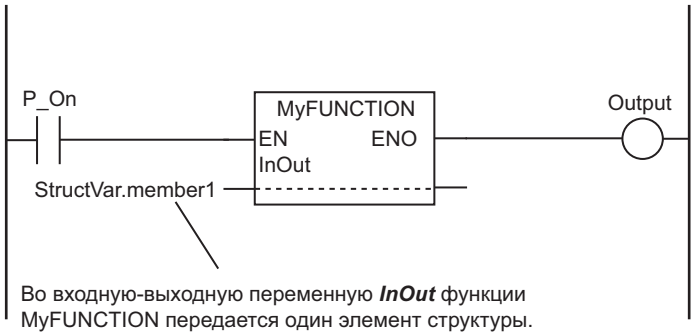
В следующем примере во входную-выходную переменную функции или функционального блока передается один член структуры, поэтому команду нельзя использовать внутри функции или функционального блока.

Переменная, вызываемая функцией MyFUNCTION

Переменная	Тип данных
StructVar	STRUCT

Переменная в функции MyFUNCTION

Переменная	Тип данных
InOut	DINT



Настройка параметра для обнаружения ошибки

Как было описано выше, в некоторых случаях команду можно использовать, даже если отображается сообщение об ошибке. В Sysmac Studio имеется параметр, с помощью которого можно включить или отключить обнаружение этой ошибки.
Чтобы исключить обнаружение данной ошибки, перейдите к настройке параметров в Sysmac Studio и снимите флажок **Обнаруживать ошибку при передаче входной-выходной переменной в определенный аргумент команды** в области **Проверка программы**. Конкретный порядок действий см. в документе *Sysmac Studio, версия 1 — Руководство по работе (Cat. No. W504)*.
Однако, прежде чем отключать обнаружение этой ошибки, убедитесь, что все команды в пользовательской программе могут использоваться. Даже если обнаружение этой ошибки будет отключено, то же самое сообщение будет отображаться в качестве предупреждения.

⚠ Осторожно!

Отключение данного параметра может привести к работе команд непредусмотренным образом, что может повлиять на систему.
Прежде чем снимать флажок для данного параметра, обязательно убедитесь в выполнении условий использования команд, которые описаны в разделе *Когда сообщение об ошибке может быть проигнорировано* на стр. A-53.

Информация о версии

Это сообщение об ошибке отображается и вышеуказанный параметр доступен только для Sysmac Studio версии 1.02.

I

Предметный указатель

Предметный указатель

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

W

X

Z

А

Б

В

I

Г

Д

Е

З

И

К

Л

М

Н

О

П

P

С

Т

У

Ф

Ц

Ч

Э

Россия
ООО «Омрон Электроникс»
Ленинградский проспект, д. 15, стр. 18, 4 этаж,
Москва, Россия, 125040
Тел.: +7 495 648 94 50
Эл. почта: omron-russia@omron.com
www.industrial.omron.ru

Информация, содержащаяся в настоящем руководстве, может быть изменена без предупреждения.
Предварительная версия. Замечания и предложения просьба направлять по адресу: omron-russia@omron.com
Cat. No. W502-RU1-32