Allen-Bradley

Общая методика для программируемых контроллеров Logix5000™

1756 ControlLogix®
1769 CompactLogix™
1789 SoftLogix™
1794 FlexLogix™, PowerFlex
700S with DriveLogix

Руководство по программированию

Rockwell Automation

Важная информация для пользователя

Эксплуатационные характеристики полупроводникового оборудования отличаются от характеристик электромеханического оборудования. В публикации SGI-1.1 фирмы Allen-Bradley *«Руководство по обеспечению безопасности при использовании, установке и обслуживании полупроводниковых устройств управления»*, имеющейся в вашем местном представительстве Rockwell Automation, а также в Интернете по адресу htpp://www.ab.com/manuals/gi, описываются некоторые важные различия между полупроводниковым оборудованием и электромеханическими устройствами с жесткими соединениями. В связи с этими различиями, а также большим разнообразием применений полупроводникового оборудования, все лица, ответственные за использование такого оборудования, должны удостовериться в приемлемости всякого предполагаемого применения такого оборудования.

Rockwell Automation ни в коем случае не отвечает за косвенный ущерб, связанный с использованием такого оборудования.

Примеры и схемы приводятся в данном руководстве исключительно для иллюстрации. Поскольку каждое конкретное оборудование характеризуется множеством специфических параметров и требований, Rockwell Automation, Inc. не берет на себя ответственность за фактическое использование продуктов на основе таких примеров и схем.

Rockwell Automation, Inc. не несет патентную ответственность в связи с использованием информации, цепей, оборудования или программного обеспечения, описанных в данном руководстве.

Воспроизведение содержания данного руководства, целиком или частично, без письменного разрешения Rockwell Automation, Inc. запрещается.

В настоящем документе используются примечания, обращающие ваше внимание на вопросы безопасности.

ПРЕДУПРЕЖДЕНИЕ 	Обозначает информацию о способах действий или обстоятельствах, которые могут привести к взрыву в опасных условиях, что может повлечь травмы или смерть людей, материальный ущерб или экономические потери.
ВАЖНО	Обозначает информацию, имеющую критическое значение для успешного применения и понимания продукта.
ВНИМАНИЕ 	Обозначает информацию о способах действий или обстоятельствах, которые могут привести к травмам или смерти людей, материальному ущербу или экономическим потерям. Такие примечания помогут вам: • обнаружить опасность • избежать опасность • понять последствия
ОПАСНОСТЬ ПОРАЖЕНИЯ ТОКОМ 	Такие знаки могут быть нанесены снаружи или внутри устройства для предупреждения о возможном наличии опасного напряжения.
ОПАСНОСТЬ ОЖОГА 	Такие знаки могут быть нанесены снаружи или внутри устройства для предупреждения о том, что поверхности могут иметь опасную температуру.

Введение

В этой версии данного документа содержится новая и обновленная информация. Такая информация обозначается вертикальной полосой, как показано слева от данного абзаца.

Обновленная информация

Данный документ содержит следующие изменения:

Раздел:	Изменение:	Стр.:
Описание типа данных, определяемого пользователем	Используйте сквозной перенос описаний для сокращения времени документирования проекта.	3-21
Назначение приоритетов периодических и событийных задач	Внесены исправления в пример того, как задачи прерывают друг друга	4-7
Выбор триггера для событийной задачи	Добавление триггера потребляемого тега в контроллеры CompactLogix, FlexLogix и DriveLogix	4-20
Экспорт/импорт релейной логики	Создайте файл, содержащий релейную логику, теги, типы данных, значения параметров и документацию для конкретной функции, операции или процесса	8-14
Разработка процедуры обработки ошибки	• В данный раздел включена информация по обработчику включения питания. Это было сделано для пояснения ситуации возникновения основной ошибки при включении питания и ее обработки при необходимости. • Пояснение по ошибке, возникающей из-за изменения режима	15-1
Устранение основной ошибки в процессе предварительного сканирования	Теперь некоторые контроллеры в процессе предварительного сканирования автоматически устраняют ошибку, вызванную тем, что индекс массива находится за границами массива.	15-8
Создание основной ошибки, определяемой пользователем	При создании основной ошибки, определяемой пользователем, для кода ошибки используйте значение в диапазоне от 990 до 999. Эти коды зарезервированы для ошибок, определяемых пользователем.	15-13
Коды не основных ошибок	Внесены исправления в недостающие коды ошибок для инструкций GSV/SSV	16-4
Выбор контроллера с энергонезависимой памятью	Добавлены следующие контроллеры: • CompactLogix5331 • CompactLogix5332E • CompactLogix5335CR • ControlLogix5560M03SE • DriveLogix5730	17-3
Оценка информации об использовании памяти в режиме оффлайн	Оценка свободной и используемой памяти контроллера в режиме оффлайн	19-2
Просмотр информации об использовании памяти в режиме выполнения	Просмотр данных о свободной и используемой памяти контроллера в процессе его работы	19-3

Назначтение данного руководства

Данный документ обеспечивает руководство при разработке проектов для контроллеров Logix5000™. Он содержит общие для всех контроллеров Logix5000 пошаговые методики выполнения следующих задач:

- Организация задач, программ и процедур
- Организация тегов
- Создание последовательной функциональной схемы
- Программирование процедур с использованием таких средств программирования, как релейная логика, функциональная блок-схема, последовательная функциональная схема или структурированный текст
- Обмен данными с другими контроллерами
- Получение/передача и обработка информации в кодах ASCII
- Обработка ошибок

Термин *«контроллер Logix5000»* относится ко всем контроллерам, использующим операционную систему Logix5000, таким как:

- Контроллеры CompactLogix™
- Контроллеры ControlLoxic®
- Контроллеры DriveLogix™
- Контроллеры FlexLogix™
- Контроллеры SoftLogix5800™

Это руководство должно использоваться вместе с руководствами пользователя для используемого вами конкретного типа контроллера. Руководства пользователя направлены на решение следующих задач:

- Установка и конфигурирование устройств ввода/вывода
- Обмен данными с устройствами по различным сетям
- Поддержание рабочего состояния батареи

Для кого предназначено это руководство

Данное руководство предназначено для лиц, занимающихся программированием приложений, использующих контроллеры Logix5000, а именно:

- инженеров-программистов
- инженеров по системам управления
- инженеров по прикладным системам
- техников КИП

Когда следует пользоваться данным руководством

Используйте это руководство при проведении следующих работ:

- разработка основной программы для вашего приложения
- внесение изменений в существующее приложение
- изолированное тестирование вашего приложения

При интеграции вашего приложения с устройствами ввода/вывода, контроллерами и сетями вашей системы:

- Обращайтесь к руководству пользователя для используемого вами конкретного типа контроллера.
- При необходимости используйте данное руководство в качестве справочника.

Как пользоваться данным руководством

Это руководство подразделяется на основные задачи, выполняемые вами при программировании контроллера Logix5000.

- Каждая глава посвящена одной задаче.
- Задачи представлены в той последовательности, в которой они обычно выполняются.

При использовании этого руководства вам встретятся некоторые термины, выделенные из остального текста:

Текст:	Означает:	Например:	Означает:
Выделенный курсивом	название элемента, который вы видите на экране или в примере	Щелкните правой кнопкой мыши по *User-Defined*…	Следует щелкнуть правой кнопкой мыши по элементу, который называется User-Defined.
Выделенный жирным шрифтом	статью Глоссария	Введите **имя**...	Если вы хотите получить дополнительную информацию, обратитесь к статье **имя** в Глоссарии
`courier`	информацию (параметр), которую вы должны ввести для своего приложения	Щелкните правой кнопкой мыши по `name_of_program`…	Вы должны указать конкретную программу в вашем приложении. Как правило, это заданное вами имя или параметр.
В квадратных скобках	клавиша на клавиатуре	Нажмите [Enter].	Нажмите на клавишу Enter.

Глава1

Начало работы

Глава 2

Обмен данными с модулями ввода/вывода

Глава 3

Организация тегов

Глава 4

Управление многозадачным режимом

Глава 5

Разработка последовательной функциональной схемы

Глава 6

Программирование последовательной функциональной схемы

Глава 7

Программирование на языке структурированного текста

Глава 8

Программирование на языке релейной логики

Глава 9

Программирование на языке функциональных блоков

Глава 10

Связь с другими устройствами

Глава 11

Создание больших массивов

Глава 12

Связь с устройствами ASCII

Глава 13

Обработка символов ASCII

Глава 14

Форсировка элементов релейной логики

Глава 15

Обработка основной ошибки

Глава 16

Отслеживание неосновных ошибок

Глава 17

Сохранение и загрузка проекта с помощью энергонезависимой памяти

Глава 18

Обеспечение защиты проекта

Глава 19

Получение информации о памяти контроллера

Приложение A

Управление многочисленными сообщениями

Приложение B

Отправка сообщения нескольким контроллерам

Приложение C

Соответствие IEC61131-3

Начало работы

Использование этой главы

Эта глава содержит начальные сведения, которые помогут вам начать работу над проектом для контроллера Logix5000™.

Создание проекта

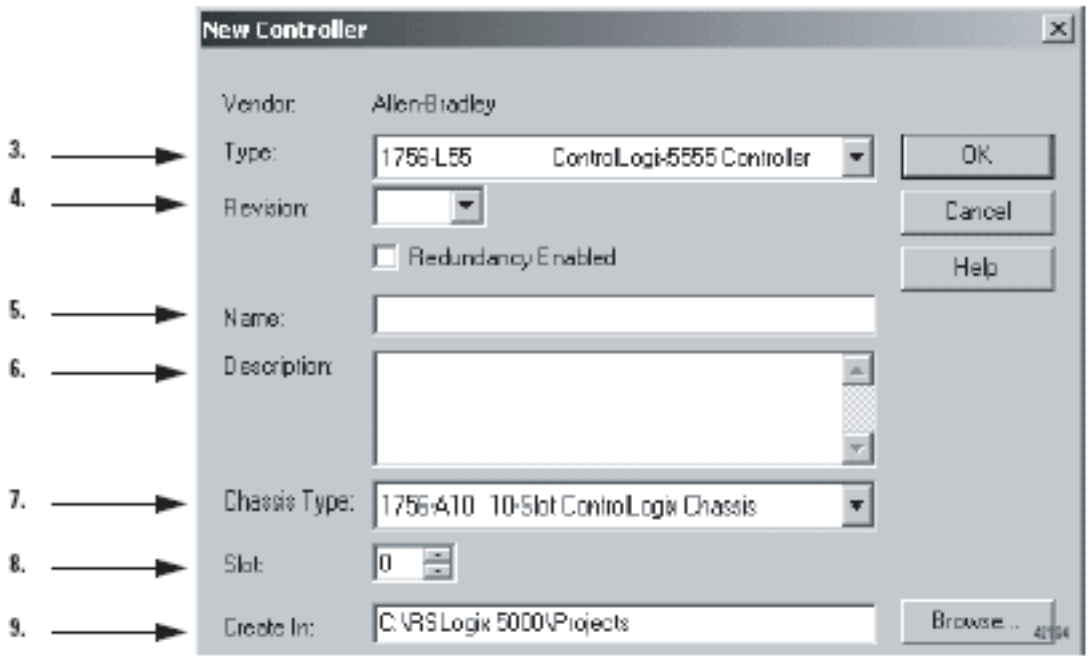
Чтобы сконфигурировать и запрограммировать контроллер Logix5000, используйте программное обеспечение RSLogix™ 5000 для создания проекта для соответствующего контроллера и управления этим проектом.

Термин:	Определение:
проект	Файл на вашей рабочей станции (или сервере), в котором хранятся логика, конфигурация, данные и документация для контроллера. • Файл проекта имеет расширение .ACD. • При создании файла проекта именем файла является имя соответствующего контроллера. • Имя контроллера не зависит от имени файла проекта. Если вы сохраните текущий файл проекта под другим именем, имя контроллера не изменится. • Если имя контроллера отличается от имени файла проекта, в строке заголовка программного обеспечения RSLogix 5000 будут показаны оба имени.

Создание проекта

1. Запустите программу RSLogix 5000.

2. В меню File (Файл) выберите New (Новый).



3. Выберите тип контроллера.

4. Выберите основную ревизию микропрограммного обеспечения для данного контроллера.

5. Введите имя (**name**) контроллера.

6. Введите описание выполняемых контроллером операций (по желанию).

7. Выберите тип шасси (количество слотов), содержащего данный контроллер (неприменимо для некоторых контроллеров).

8. Выберите или введите номер слота, в котором установлен данный контроллер (неприменимо для некоторых контроллеров).

9. Для сохранения файла в другой папке (отличной от пути по умолчанию *Create In* (Создать в)) нажмите *Browse* (Просмотр) и выберите папку.

10. Нажмите OK

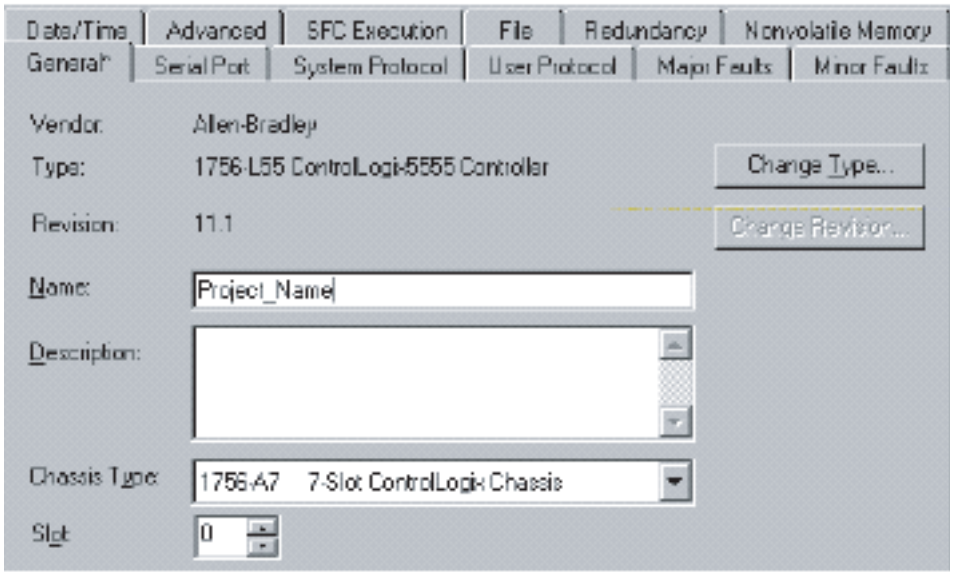
Имена:

- должны содержать только буквенные знаки (A-Z или a-z), цифровые знаки (0-9) и знаки подчеркивания (_)
- должны начинаться с буквенного знака или знака подчеркивания
- должны содержать не более 40 знаков
- знаки подчеркивания (_) не должны располагаться подряд или в конце строки
- не чувствительны к регистру

Конфигурирование проекта

Чтобы изменить конфигурацию контроллера, например, его имя, размер шасси или номер слота, используйте диалоговое окно Controller Properties (Свойства контроллера).

1. На панели инструментов режима онлайн нажмите на кнопку свойств контроллера.

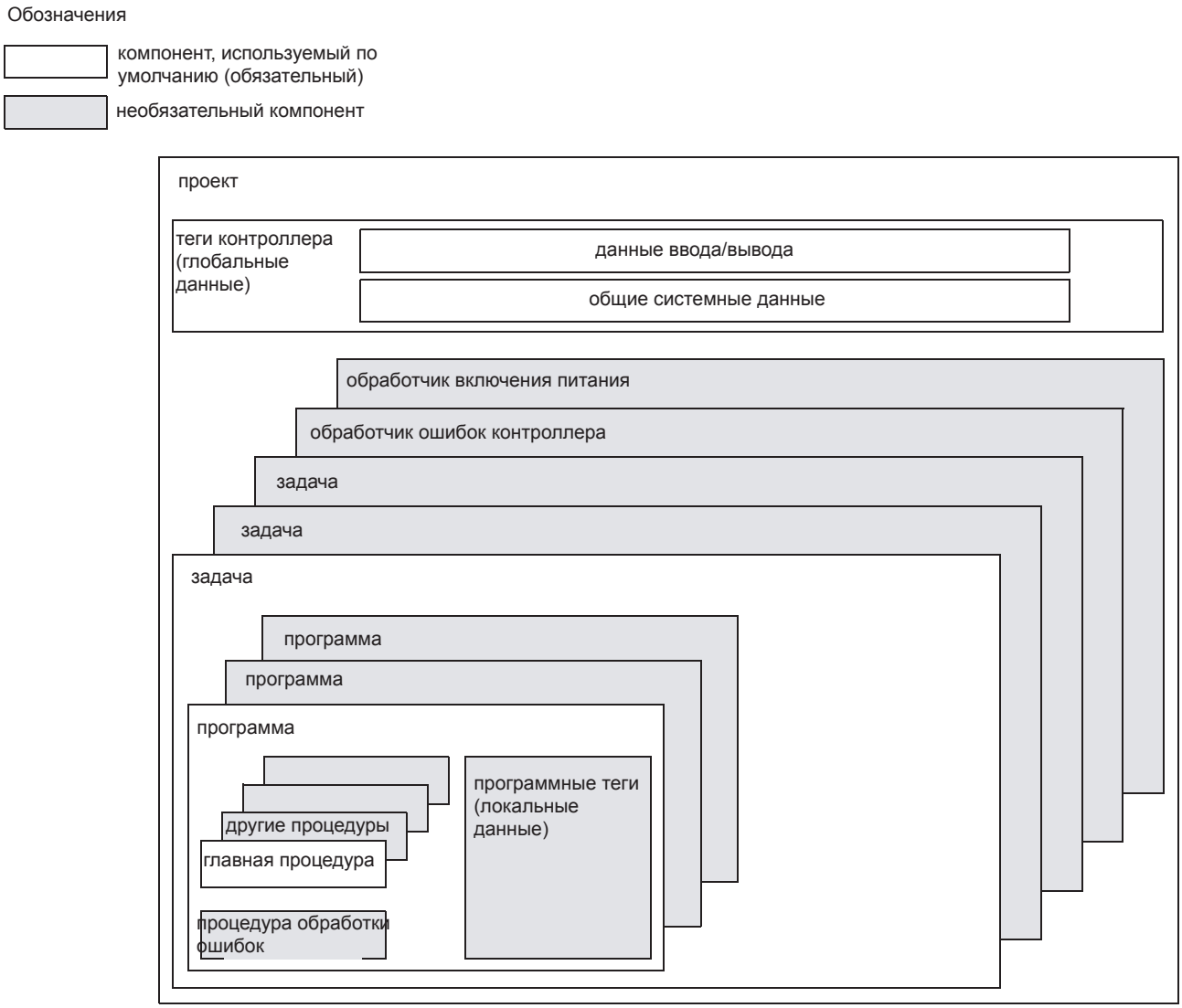


2. Внесите необходимые изменения.

3. Нажмите 

Изучение проекта

Проект включает следующие основные компоненты:

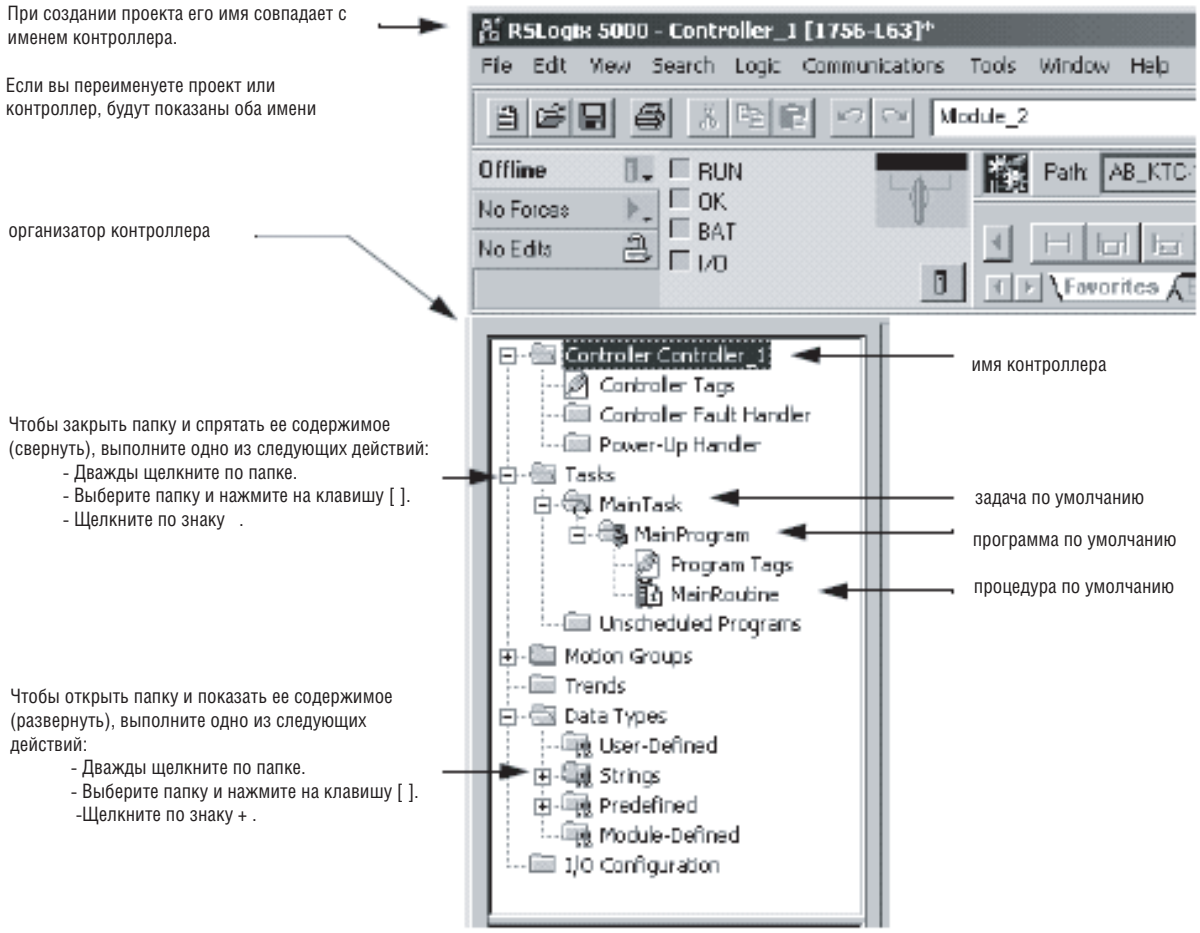


Компоненты проекта работают вместе следующим образом:

Компонент проекта:	**Определение:**
Задача	Задача предоставляет информацию о планировании и приоритетах одной или нескольким программам. Когда вы создаете новый проект, программное обеспечение RSLogix 5000 автоматически создает начальную задачу, настроенную на постоянную работу (непрерывная задача). Когда задача завершает полное сканирование, она немедленно перезапускается.
Программа	Для каждой задачи требуется как минимум одна программа. • Задача может иметь до 32 отдельных программ, каждая из которых имеет свои программные теги, главную процедуру и другие процедуры, а также необязательную процедуру обработки ошибок. • После запуска (активации) задачи все назначенные (запланированные) для данной задачи программы выполняются в том порядке, в котором они показаны в организаторе контроллера. • Вы планируете выполнение программы лишь в одной задаче и не можете разделять программу с несколькими задачами.
Процедура	Процедуры обеспечивают исполняемый код для проекта в контроллере (аналогично программному файлу в контроллере PLC или SLC). Каждая процедура использует определенный язык программирования, например, релейную логику.
Главная процедура	Когда программа выполняется, в первую очередь выполняется ее главная процедура. Используйте главную процедуру для вызова (выполнения) других процедур (подпрограмм). Для вызова другой процедуры из программы, используйте инструкцию Jump to Subroutine (JSR).
Подпрограмма	Всякая процедура, не являющаяся главной процедурой и процедурой обработки ошибок. Для выполнения подпрограммы используйте инструкцию Jump to Subroutine (JSR) в другой процедуре, например, в главной процедуре.

Организатор контроллера

В программном обеспечении RSLogix 5000 организатор контроллера обеспечивает графическое представление проекта. Когда вы создаете проект, RSLogix 5000 автоматически создает задачу, программу и процедуру по умолчанию.



Создание процедур

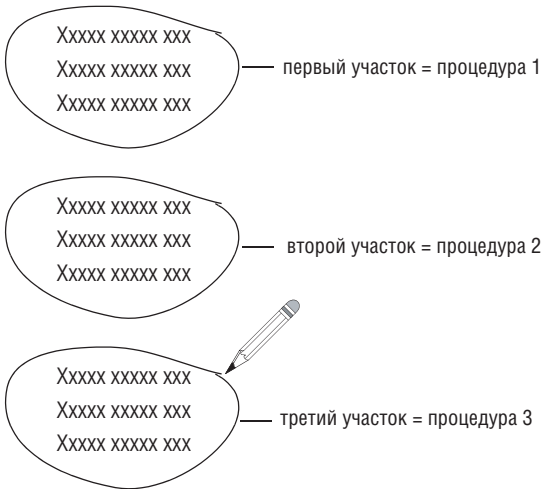
Процедуры обеспечивают исполняемый код для проекта в контроллере.

Задание процедуры для каждого участка вашей установки или процесса

Для упрощения разработки, тестирования и отладки вашего проекта разделите его на процедуры (подпрограммы).

1. Выделите каждый физический участок вашей установки или процесса.
2. Назначьте процедуру каждого из таких участков.

Описание вашей установки или процесса



Определение установленных языков программирования

Чтобы определить, какие языки программирования установлены в вашей версии программного продукта RSLogix 5000:

1. Запустите RSLogix 5000.
2. В меню *Help* (Справка) выберите *About RSLogix 5000* (Об RSLogix 5000).

За инструкциями по добавлению языка программирования обращайтесь к публикации 1756-SG001 *«Руководство по выбору ControlLogix»*.

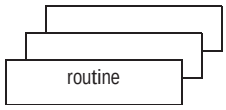
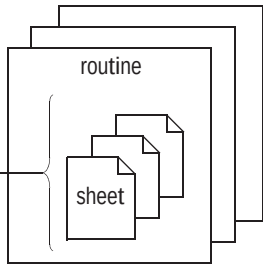
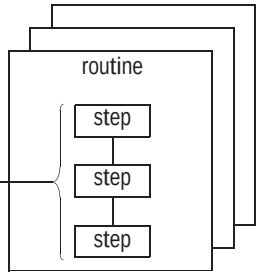
Выбор языка программирования для каждого участка

Выберите подходящий язык программирования для каждого участка вашей установки или процесса.

- Контроллеры Logix5000 позволяют использовать следующие языки:
  - релейная логика
  - функциональная блок-схема
  - последовательная функциональная схема
  - структурированный текст
- Используйте любую комбинацию этих языков в одном проекте.

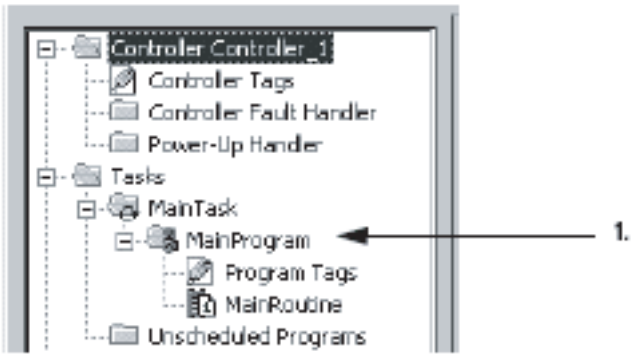
Как правило, если участок вашего кода представляет собой:	**Используется этот язык:**
непрерывное или параллельное выполнение нескольких операций (не последовательных)	релейная логика
булевы или побитовые операции	
сложные логические операции	
обработку сообщений и передачи данных	
блокировку механизма	
операции, которые при необходимости должны интерпретироваться обслуживающим персоналом для диагностики установки или процесса	
управление непрерывным процессом и приводом	функциональная блок-схема
управление циклом	
расчеты в схеме	
управление несколькими операциями высокого уровня	последовательная функциональная схема (ПФС)
повторяемые последовательности операций	
групповую обработку	
управление перемещением с помощью структурированного текста	
операции конечного автомата	
сложные математические операции	структурированный текст
обработку специального массива или таблицы в цикле	
обработку строк ASCII или протоколов	

Разбивка каждой процедуры на более осмысленные части

Если процедура использует этот язык:	**То:**	**Пример:**
релейная логика структурированный текст	Разбейте большие процедуры на несколько более мелких 	Для непрерывного выполнения нескольких сложных булевых операций... ... создайте отдельную процедуру для каждой операции.
функциональная блок-схема (ФБС)	Внутри процедуры ФБС создайте отдельный лист для каждого функционального цикла каждого устройства (двигателя, клапана и т.п.). 	Для управления 4 клапанами, при котором каждому клапану требуется обратная связь с подтверждением, что он находится в заданном положении, ... создайте отдельный лист для каждого клапана.
последовательная функциональная схема (ПФС)	Разбейте ПФС на шаги. 	Для выполнения следующей последовательности: 1. Заполнение резервуара 2. Смешивание ингредиентов в резервуаре. 3. Опорожнение резервуара... ... сделайте каждый этап (заполнение, смешивание, опорожнение) отдельным шагом.

Создание процедуры

Для каждой программы требуется как минимум одна процедура. Используйте процедуру для выполнения вышей логики на определенном языке программирования.



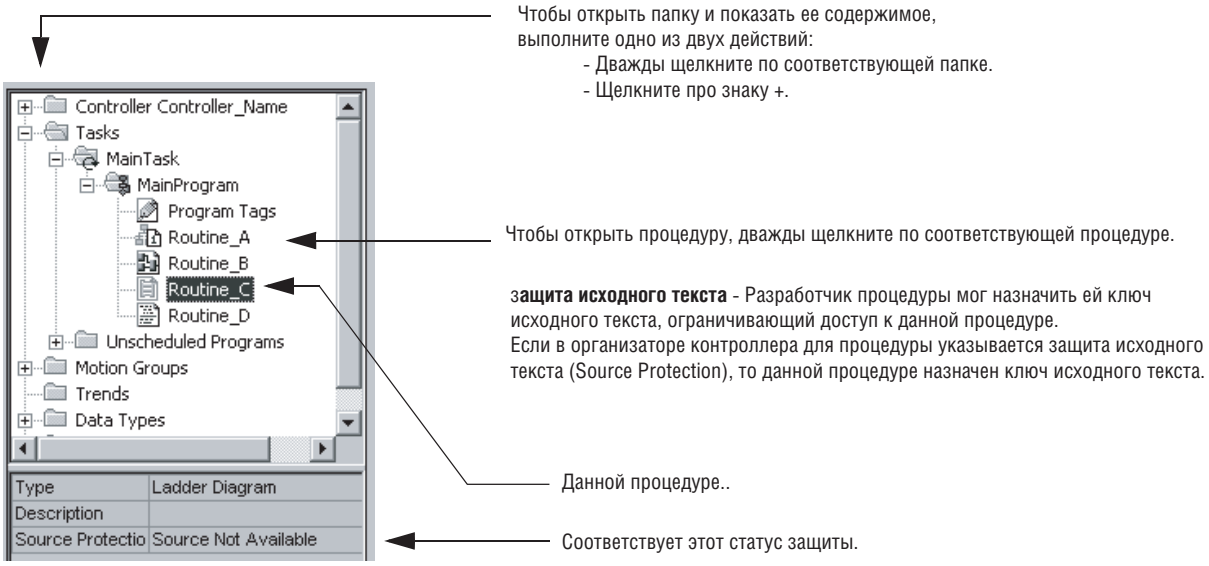
1. В организаторе контроллера щелкните правой кнопкой мыши по программе, которая будет выполнять процедуру, и выберите *New Routine* (Новая процедура).

2. В текстовом окне *Name* (Имя) введите **имя** данной процедуры.

3. В списке *Type* (Тип) выберите язык программирования для данной процедуры.

4. Нажмите 

Откртытие процедуры



Если статус защиты:	**То:**
Source Not Available (Исходный текст недоступен)	Чтобы открыть данную процедуру, вашему компьютеру требуется ключ исходного текста для нее.
Source Not Available (Viewable) (Исходный текст недоступен (Может просматриваться))	• Вы можете лишь открыть и просмотреть данную процедуру. • Вы *не можете* вносить какие-либо изменения или копировать какую-либо часть процедуры.
Source Available (Исходный текст доступен)	Вы имеете полный доступ к данной процедуре.
Source Available (Viewable) (Источник доступен (Может просматриваться))	Вы имеете полный доступ к данной процедуре.

ВАЖНО

Если исходный текст процедуры недоступен, *не* экспортируйте проект.

- Файл экспорта (.L5K) содержит лишь те процедуры, для которых доступен исходный код.
- Если вы экспортируете проект, в котором исходный код доступен *не* для всех процедур, вы *не* сможете восстановить весь проект.

СОВЕТ

Если процедура не открывается, возможно, на вашем компьютере не установлен необходимый язык программирования.

- Чтобы выяснить, какие языки программирования установлены на вашем компьютере, выберите *Help* (Справка) ⇒ *About RSLogix 5000* (Об RSLogix 5000).
- За инструкциями по добавлению языка программирования обращайтесь к публикации 1756-SG001 «Руководство по выбору ControlLogix».

Проверка проекта

В процессе программирования своего проекта периодически проверяйте свою работу:

1.. В самой верхней панели инструментов окна RSLogix 5000 нажмите на кнопку 

2. Если внизу окна появятся какие-либо ошибки:

a Чтобы перейти к первой ошибке или предупреждению, нажмите [F4].

b. Исправьте ошибку в соответствии с описанием в окне Results (Результаты).

c. Перейдите к шагу 1.

3. Чтобы закрыть окно Results (Результаты), нажмите [Alt] + [1].

Сохранение проекта

Сохраняйте проект в процессе создания логики и внесения изменений в конфигурацию.

Чтобы:	**Выполните следующее:**
сохранить внесенные изменения	Выберите *Save* (Сохранить)в меню *File* (Файл).
сделать копию открытого проекта, оставив прежним имя контроллера	1. Выберите *Save As* (Сохранить как) в меню *File* (Файл). 2. Введите имя файла проекта. Вместо пробелов используйте знаки подчеркивания [_]. 3. Нажмите *Save* (Сохранить).
сделать копию проекта и присвоить контроллеру другое имя	1. Выберите *Save As* (Сохранить как) в меню *File* (Файл). 2. Введите имя файла проекта. Вместо пробелов используйте знаки подчеркивания [_]. 3. Нажмите *Save* (Сохранить). 4. В организаторе контроллера щелкните правой кнопкой мыши по папке *Controller* `name_of_controller` и выберите *Properties* (Свойства). 5. Введите новое имя контроллера. 6. Нажмите OK.

Если вы вносите изменения в проект в **режиме онлайн**, сохраните проект таким образом, чтобы оффлайновый файл проекта соответствовал онлайновому файлу проекта:

Если вы хотите:	**Выполните следующее:**
сохранить онлайновые изменения *и* данные	Выберите *Save* (Сохранить)в меню *File* (Файл).
сохранить онлайновые изменения, *не* сохраняя онлайновые данные	1. В меню *Communications* (Связь) выберите *Go Offline* (Перейти в режим оффлайн). 2. Выберите *Save* (Сохранить)в меню *File* (Файл).

Конфигурирование драйвера связи

Программному обеспечению RSLogix 5000 требуется драйвер связи для обмена данными с контроллером. Драйверы связи конфигурируются с помощью программного продукта RSLinx®:

1. Запустите RSLinx.
2. В меню *Communications* (Связь) выберите *Configure Drivers* (Сконфигурировать драйверы).
3. В выпадающем списке Available Driver Types (Имеющиеся типы драйверов) выберите *Configure Drivers* (Сконфигурировать драйверы).

Для этой сети:	И этого типа компьютера:	Выберите этот драйвер:
последовательная	→	RS-232 DF1 Devices
DH+™	настольный компьютер	17-84-KT/KTX(D)/PKTX(D)
	портативный компьютер	1784-PCMK
ControlNet™	настольный компьютер	1784-KTC(X)
	портативный компьютер	1784-PCC
EtherNet/IP	→	Ethernet devices
DeviceNet™	→	DeviceNet Drivers (драйверы 1784-PCD/PCIDS, 1770-KFD, SDNPT)

4. Выберите *Add New* (Добавить новый).
5. Если вы хотите присвоить драйверу описательное имя, измените имя, заданное по умолчанию.
6. Нажмите ОК.
7. Сконфигурируйте драйвер:

Для этого драйвера:	Выполните следующее:
последовательный	А. В выпадающем списке *Comm Port* (Коммуникационный порт) выберите последовательный порт, который будет использоваться данным драйвером. Б. В выпадающем списке Device (Устройство) выберите *Logix 5550 Serial Port*. В. Нажмите *Auto-Configure* (Автоконфигурирование).
ControlNet	А. В поле *Station Name* (Имя станции) введите имя, которое будет идентифицировать данный компьютер в окне RSWho. Б. Выберите значение прерывания (interrupt value), адрес памяти (memory address) и базовый адрес ввода/вывода (I/O base address). В. В поле *Net Address* (Сетевой адрес) введите номер узла ControlNet, который вы хотите назначить данному компьютеру.
DH+	А. В выпадающем списке *Value* (Значение) выберите тип интерфейсной платы, который будет использоваться данным драйвером. Б. В выпадающем списке *Properties* (Свойства) выберите следующий пункт. В. В поле *Value* (Значение) введите или выберите соответствующее значение. Г. Повторите шаги Б и В для остальных свойств.
Ethernet	Для каждого устройства Ethernet в данной сети, с которым вы хотите обмениваться данными (например, для каждого модуля 1756-ENB или контроллера PLC-5E) добавьте запись в таблицу соответствий: А. В столбце *Host Name* (Имя хоста) введите IP-адрес или имя хоста для данного устройства Ethernet. Б. Для обмена данными с другим устройством Ethernet в этой сети выберите *Add New* (Добавить новый) и перейдите к шагу А.

8. Нажмите ОК, затем *Close* (Закрыть).

Загрузка проекта в контроллер

Используйте эту процедуру для загрузки проекта в контроллер, чтобы можно было выполнить его логику.

- При загрузке проекта вы теряете проект и данные, находящиеся в этот момент в контроллере.
- Если ревизия контроллера не соответствует ревизии проекта, система попросит вас обновить микропрограммное обеспечение контроллера. Программа RSLogix 5000 позволяет обновлять микропрограммное обеспечение в процессе загрузки проекта.

ВНИМАНИЕ

При загрузке проекта или обновлении микропрограммного обеспечения все оси сервосистемы отключаются. Прежде чем загружать проект или обновлять микропрограммное обеспечение, убедитесь в том, что это *не* приведет к какому-либо непредвиденному перемещению оси.

ВАЖНО

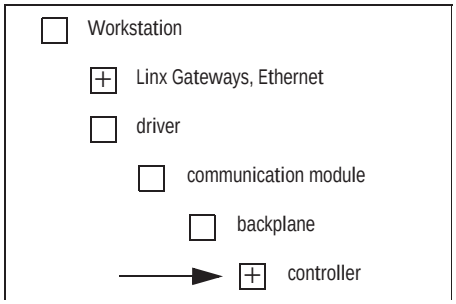
Для обновления микропрограммного обеспечения сначала установите комплект модернизации микропрограммного обеспечения.

- Пакет модернизации поставляется вместе с программным обеспечением RSLogix 5000 на дополнительном компакт-диске.
- Пакет модернизации можно скачать с сайта www.ab.com. Выберите *Product Support* (Поддержка продуктов), затем *Firmware Updates* (Обновления микропрограммного обеспечения).

1. Откройте проект RSLogix 5000, который вы хотите загрузить.
2. В меню Communications (Связь) выберите *Who Active* (Кто активен).
3. Расширяйте сеть до тех пор, пока вы не увидите контроллер.

Чтобы расширить сеть на один уровень, выполните одно из следующих действий

- Дважды щелкните по сети.
- Выберите сеть и нажмите на клавишу —>.
- Щелкните по знаку +.



4. Выберите контроллер.
5. Выберите *Download* (Загрузить).

6. Проверьте ответ программного обеспечения:

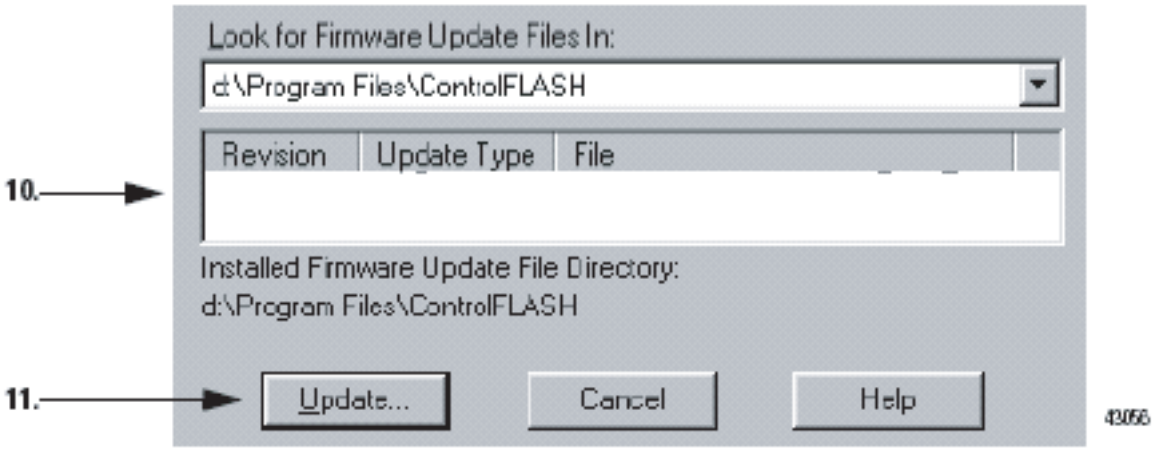
Если программное обеспечение укажет:	**То:**
Download to the controller (Загрузить в контроллер)	Перейдите к шагу 7.
Failed to download to the controller. The revision of the offline project and controller's firmware are not compatible. (Загрузка в контроллер не прошла. Ревизия оффлайнового проекта несовместима с микропрограммным обеспечением контроллера)	Перейдите к шагу 9.

7. Выберите *Download* (Загрузить).

Проект загружается в контроллер, а RSLogix 5000 переходит в режим онлайн.

8.. Пропустите оставшуюся часть данной процедуры.

9. Выберите *Update Firmware* (Обновить микропрограммное обеспечение)



10. Выберите необходимую ревизию контроллера.

11. Выберите *Update* (Обновить).

Появится диалоговое окно с запросом подтверждения обновления.

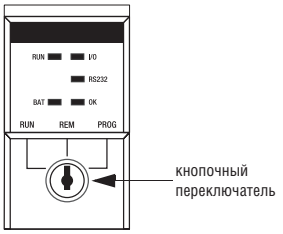
12. Для обновления контроллера выберите *Yes*.

Произойдут следующие события:

- Микропрограммное обеспечение контроллера обновится.
- Проект загрузится в контроллер.
- RSLogix 5000 перейдет в режим онлайн.

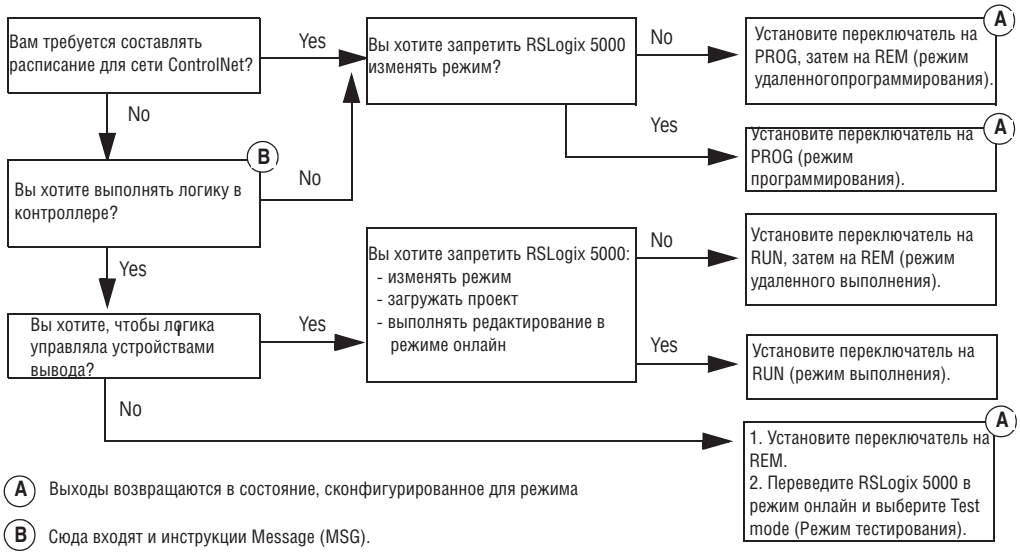
Выбор режима для контроллера

Для изменения режима работы контроллера используйте кнопочный переключатель, расположенный на передней панели контроллера:



ВАЖНО

- Во всех режимах происходит получение и отправка данных в ответ на сообщение от другого контроллера.
- Во всех режимах производятся и потребляются теги.



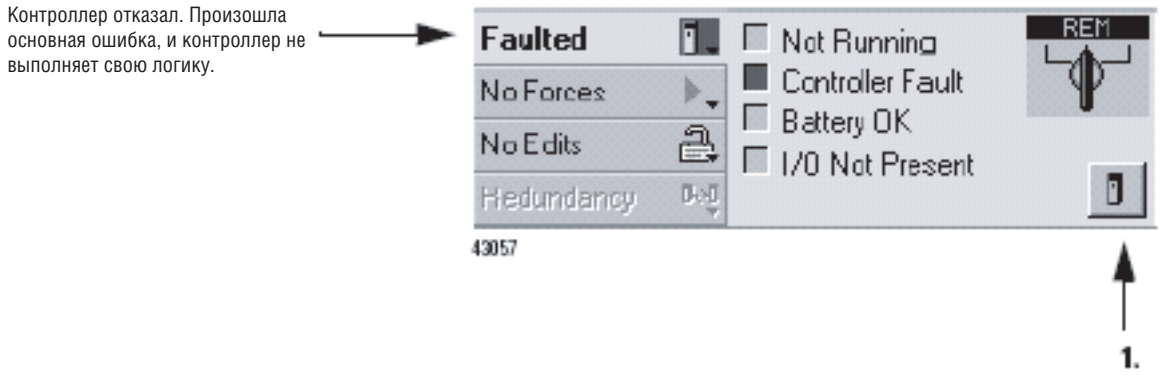
Вы также можете изменить режим контроллера при помощи программного обеспечения RSLogix 5000:

1. Установите кнопочный переключатель на передней панели контроллера в положение REM.
2. Переведите контроллер в режим онлайн.

3. Выберите нужный вам режим работы контроллера на панели инструментов режима онлайн.

Ручной сброс основной ошибки

Если контроллер войдет в **режим ошибки**, это означает, что возникла **основная ошибка**, и контроллер прекратил выполнение логики.



Для исправления основной ошибки:

1. Нажмите на кнопку .
2. Используйте информацию из списка последних ошибок (*Recent faults list)* для устранения причины данной ошибки. См. раздел «Коды основных ошибок» на стр. 15-15.
3. Щелкните по кнопке *Clear Majors* (Сбросить основные ошибки).

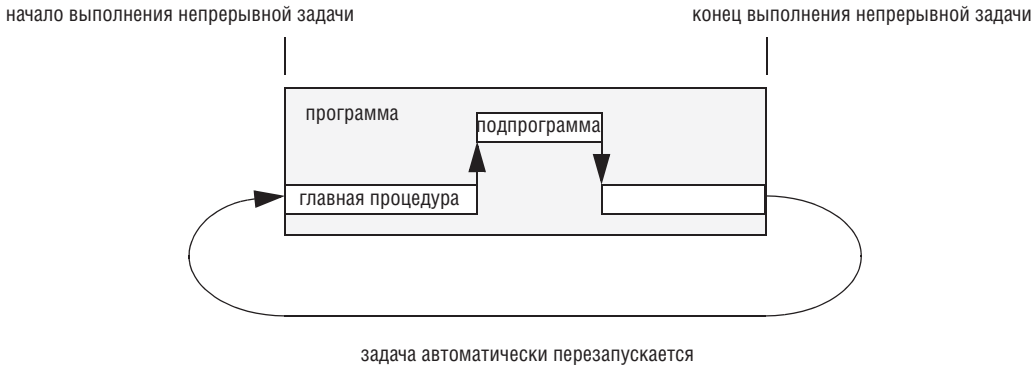
СОВЕТ

Вы также можете сбросить основную ошибку с помощью кнопочного переключателя на контроллере. Установите переключатель в положение *Prog*, затем в положение *Run*, а затем опять в положение *Prog*.

Конфигурирование выполнения задачи

Когда вы создаете новый проект, RSLogix 5000 автоматически создает начальную задачу, настроенную на постоянное выполнение (непрерывная задача). После завершения этой задачей полного сканирования, задача немедленно перезапускается.

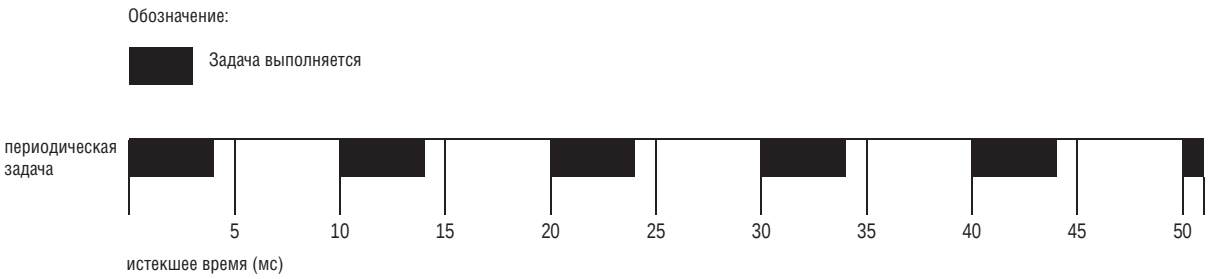
Рисунок 1.1 Выполнение непрерывной задачи



Если вы знакомы с приложением DCS или планируете программировать свою систему с помощью функциональной блок-схемы, вы можете сконфигурировать задачу таким образом, чтобы она выполнялась с определенной периодичностью (периодическая задача). Это позволяет выполнять обновление вашей функциональной блок-схемы с указанной вами периодичностью.

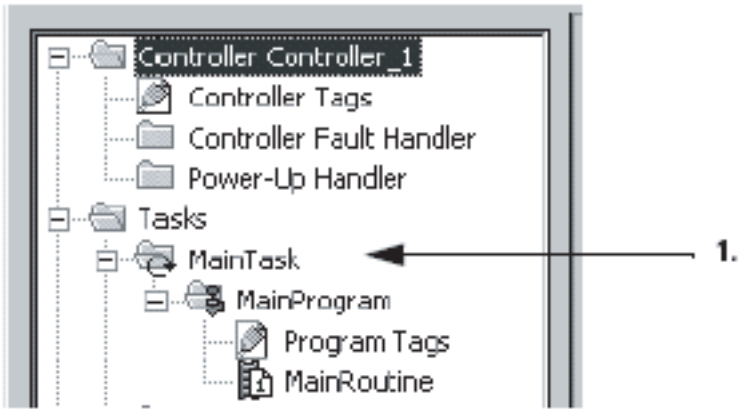
- По истечении заданного для задачи периода происходит однократное выполнение задачи.
- Вы можете задать период от 1 мс до 2000 с. Значение по умолчанию – 10 мс.
- Если вы используете периодическую задачу вместе с непрерывной задачей, периодическая задача прерывает выполнение непрерывной задачи. После завершения выполнения периодической задачи управление возвращается к непрерывной задаче. За дополнительной информацией по работе в многозадачном режиме обращайтесь к главе 4.

Рисунок 1.2 Пример периодической задачи, выполняющейся каждые 10 мс.

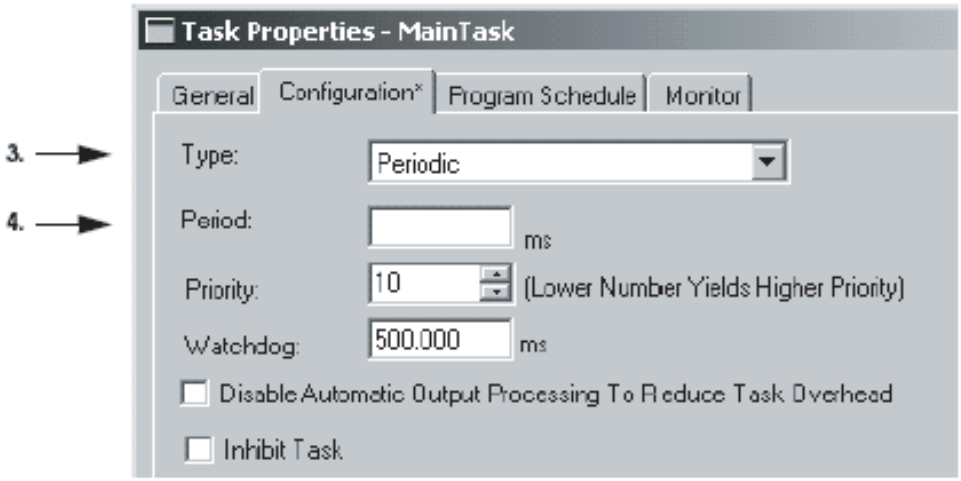


Конфигурирование задачи

Чтобы сконфигурировать выполнение задачи, используйте диалоговое окно свойств (Properties) для данной задачи.



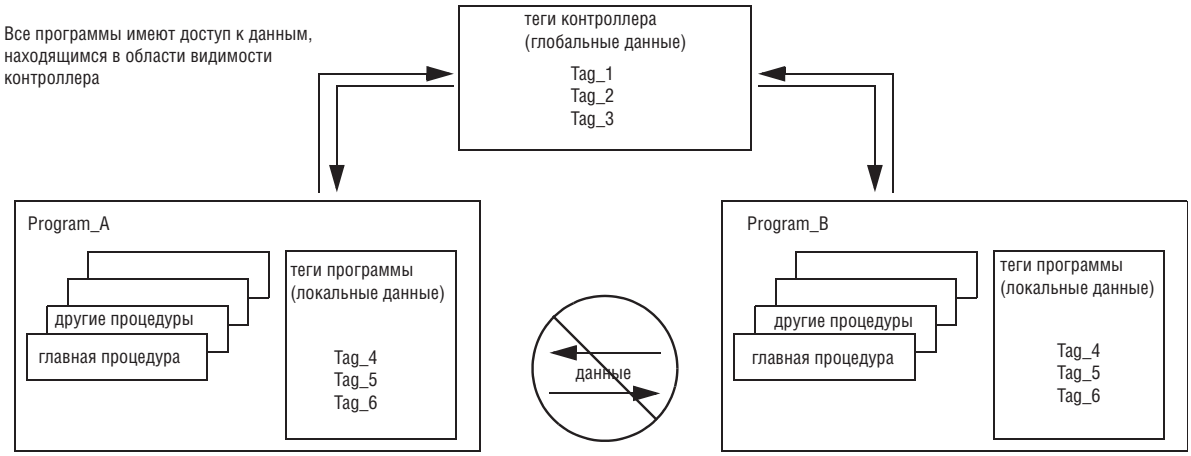
1. В организаторе контроллера щелкните правой кнопкой мыши по задаче, которую вы хотите сконфигурировать, и выберите *Properties* (Свойства).
2. Щелкните по закладке *Configuration* (Конфигурация).



3. Из списка *Type* (Тип) выберите тип выполнения для данной задачи. Разрешается иметь лишь одну непрерывную задачу.
4. Если в шаге 3 вы выбрали Periodic (Периодическая), то введите периодичность, с которой должна выполняться данная задача.
5. Нажмите 

Создание нескольких программ

Контроллер Logix5000 позволяет вам разделить ваше приложение на несколько программ, со своими данными каждая. При этом *не* требуется управлять конфликтующими именами тегов из разных программ. Это облегчает многократное использование имен кодов и тегов в нескольких программах.

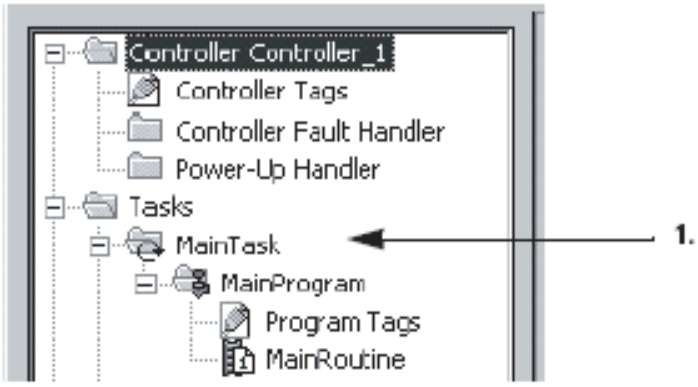


Данные в области видимости программы изолированы от других программ:

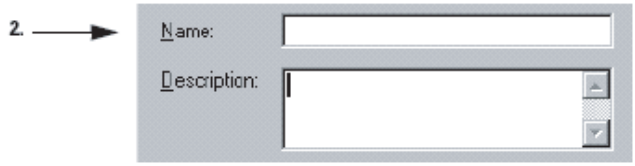
- Процедуры не могут обращаться к данным, находящимся в области видимости другой программы.
- Вы можете многократно использовать в нескольких программах имя тега, находящегося в области видимости программы. Например, обе программы (Program_A и Program_B) могут иметь тег программы с именем Tag_4.

Создание программы

Для каждой задачи требуется как минимум одна программа. Вы можете создать несколько программ для одной задачи.



1. В организаторе контроллера щелкните правой кнопкой мыши по задаче, которая будет выполнять данную программу, и выберите *New Program* (Новая программа).

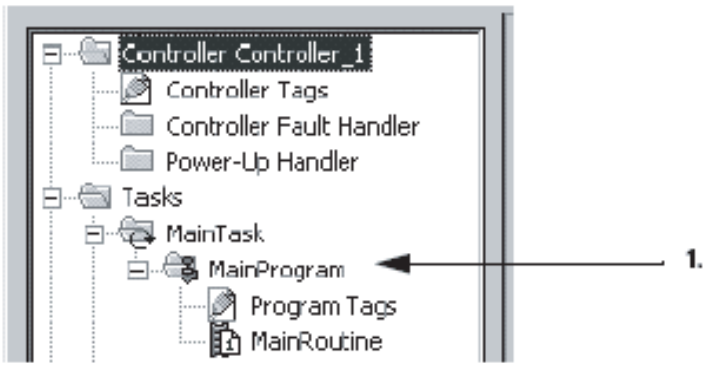


2.В текстовом поле *Name* (Имя) введите **имя** для программы.

3. Нажмите OK

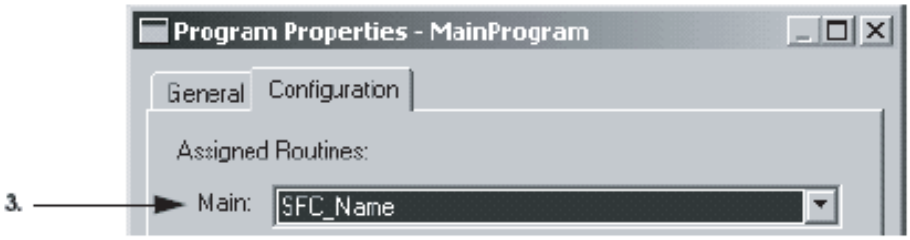
Конфигурирование программы

Для каждой программы требуется главная процедура. Главная процедура выполняется при каждом выполнении программы.



1. В организаторе контроллера щелкните правой кнопкой мыши по программе, которую вы хотите сконфигурировать, и выберите *Properties* (Свойства).

2. Щелкните по закладке *Configuration* (Конфигурация).



3. Из списка *Main* (Главная) выберите имя процедуры, которая должна выполняться в качестве главной процедуры.

4. Нажмите OK

Доступ к информации о состоянии

Контроллеры Logix5000, в отличие от контроллеров PLC-5, не имеют файла состояния. Доступ к информации о состоянии можно получить при помощи ключевого слова или обращения к определенному объекту.

Если вы хотите:	Обращайтесь к:
использовать в вашей логике определенные ключевые слова для контроля конкретных событий	разделу «Контроль флагов состояния» на стр. 1-22
получать или устанавливать системные значения	разделу «Получение и задание системных данных» на стр. 1-23

Контроль флагов состояния

Контроллер поддерживает ключевые слова состояния, которые вы можете использовать в вашей логике, чтобы контролировать определенные события:

- Ключевые слова состояния *не* чувствительны к регистру.
- Поскольку флаги состояния могут изменяться очень быстро, RSLogix 5000 *не* показывает на экране состояние этих флагов. (Т.е. даже при установленном флаге состояния инструкция, к которой относится данный флаг, *не* выделяется.)
- Вы *не можете* задать псевдоним тега для ключевого слова.

Вы можете использовать следующие ключевые слова:

Чтобы определить, имеет ли место следующее:	Используйте:
сохраняемое вами значение не подходит адресату, так как оно: • больше максимального значения, заданного для адресата, или • меньше минимального значения, заданного для адресата **Важно:** При каждом переходе S:V из сброшенного состояния в установленное оно генерирует не основную ошибку (тип 4, код 4)	S:V
значение адресата данной инструкции равно 0	S:Z
значение адресата данной инструкции является отрицательной величиной	S:N
при выполнении арифметической операции происходит перенос или заем разряда с попыткой использования битов, выходящих за пределы данного типа данных Например: • при сложении 3 + 9 происходит перенос 1 • при вычитании 25 – 18 происходит заем 10	S:C
это первое нормальное сканирование процедур в текущей программе	S:FS
была сгенерирована как минимум одна не основная ошибка: • Контроллер устанавливает этот бит при возникновении не основной ошибки из-за выполнения программы. • Контроллер не устанавливает этот бит для не основных ошибок, не связанных с выполнением программы, например, если разрядилась аккумуляторная батарея.	S:MINOR

Получение и задание системных данных

Контроллер сохраняет системные данные в объектах. Здесь нет файла состояния, как в контроллере PLC-5. Используйте инструкции GSV/SSV для получения и установки системных данных контроллера, хранящихся в объектах:

- Инструкция GSV находит указанную информацию и помещает ее по заданному адресу.
- Инструкция SSV присваивает указанному атрибуту значение, взятое из заданного источника.

ВНИМАНИЕ

С осторожностью используйте инструкцию SSV. Внесение изменений в объекты может привести к непредвиденным действиям контроллера или травмам персонала.

Для получения или задания системного значения:

1. Откройте проект RSLogix 5000.
2. В меню *Help* (Справка) выберите *Contents* (Содержание).
3. Щелкните по закладке *Index* (Указатель).
4. Введите *gsv/ssv objects* и нажмите *Display* (Показать).

5. Щелкните по требуемому объекту.

Для получения или задания данных по:	**Щелкните по:**
оси сервомодуля	AXIS
кванту времени на служебные системные операции	CONTROLLER
аппаратным средствам контроллера	CONTROLLERDEVICE
согласованному системному времени для устройств в одном шасси	CST
драйверу связи DF1 для последовательного порта	DF1
истории ошибок для контроллера	FAULTLOG
атрибутам инструкции Message	MESSAGE
статусу, ошибкам и режиму модуля	MODULE
группе осей	MOTIONGROUP
информации об ошибках или времени сканирования для программы	PROGRAM
номеру экземпляра процедуры	ROUTINE
конфигурации последовательного порта	SERIALPORT
свойствам или истекшего времени выполнения задачи	TASK
времени часов контроллера	WALLCLOCKTIME

6. В списке атрибутов для данного объекта найдите атрибут, к которому вы хотите обратиться.

7. Создайте тег для значения этого атрибута:

Если тип данных этого атрибута:	**То:**
один элемент (например, DINT)	Создайте тег для данного атрибута.
несколько элементов (например, DINT[7])	А. Создайте пользовательский тип данных, соответствующий организации данных, используемых данным атрибутом. Б. Создайте тег для данного атрибута и используйте тип данных из шага А.

8.. Введите соответствующую инструкцию в вашей процедуре релейной логики:

Для:	**Введите эту инструкцию:**
получения значения атрибута	GSV
задания значения атрибута	SSV

9. Задайте необходимые операнды для данной инструкции:

Для этого операнда:	**Выберите:**
Class name (Имя класса)	название объекта
Instance name (Имя экземпляра)	имя конкретного объекта (например, имя требуемого модуля ввода/вывода, задачи, сообщения) Ввод этого имени требуется не для всех объектов. Для обозначения текущей задачи, программы или процедуры выберите *THIS*.
Attribute name (Имя атрибута)	имя атрибута
Dest (GSV) (Приемник)	тег, в котором будет сохранено найденное значение Если этот тег представляет собой заданный пользователем тип данных или массив, то выберите его первый член или элемент.
Source (SSV) (Источник)	тег, в котором хранится значение, подлежащее установке Если этот тег представляет собой заданный пользователем тип данных или массив, то выберите его первый член или элемент.

Следующий пример показывает получение текущей даты и времени.

ПРИМЕР Получение системного значения

При первом сканировании находится атрибут *DateTime* объекта *WALLCLOCKTIME* и сохраняется в теге *wall_clock*, использующем пользовательский тип данных.

За дополнительной информацией обращайтесь к публикации1756-RM003 «Справочное руководство по общему набору инструкций контроллеров Logix5000».

Корректировка кванта времени на служебные системные операции

Контроллер Logix5000 обменивается данными с другими устройствами (модулями ввода/вывода, контроллерами, терминалами HMI и т.д.) либо с заданной периодичностью (запланированный обмен), либо при наличии времени обработки для обслуживания передачи данных (незапланированный обмен).

Обмен данными такого типа:	Является:
обновление данных ввода/вывода (кроме пересылки блоков)	Запланированным
производство или потребление тегов	
обмен с устройствами программирования (например, с пакетом RSLogix 5000)	Незапланированным
обмен с устройствами HMI	
выполнение инструкций Message (MSG), включая пересылку блоков	
ответ на сообщения от других контроллеров	
синхронизация вспомогательного контроллера резервированной системы	
восстановление и контроль соединений ввода/вывода (например, при удалении и установке модуля ввода/вывода при подключенном питании); сюда *не* входят обычные обновления ввода/вывода, осуществляемые при выполнении логики.	
создание моста для связи между последовательным портом контроллера и другими устройствами ControlLogix через объединительную плату ControlLogix	

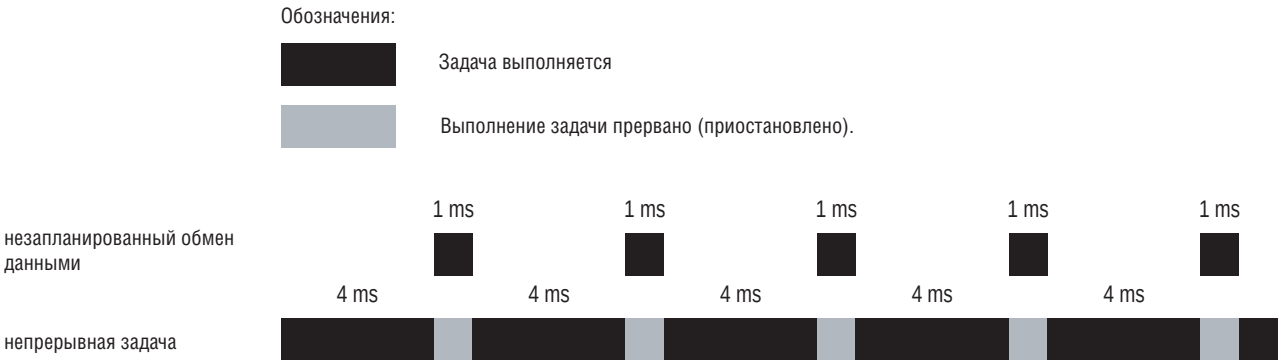
Незапланированный обмен данными – это всякий обмен данными, который вы *не* конфигурируете при помощи папки I/O configuration (Конфигурация ввода/вывода) вашего проекта.

- **Квант времени на служебные системные операции** – это доля времени (за исключением времени выполнения периодических и событийных задач), посвящаемая контроллером незапланированному обмену данными.
- Контроллер выполняет незапланированный обмен данными в течение максимум 1 мс, после чего возвращается к выполнению непрерывной задачи.

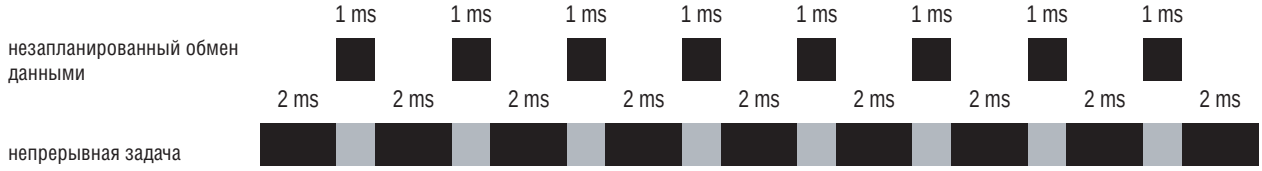
В следующей таблице показано отношение между непрерывной задачей и незапланированным обменом данными при различных квантах времени на служебные системные операции:

При таком кванте времени:	Непрерывная задача работает в течение:	А незапланированный обмен данными выполняется в течение максимум:
10%	9 мс	1 мс
20%	4 мс	1 мс
33%	2 мс	1 мс
50%	1 мс	1 мс

При кванте времени на служебные системные операции 20% незапланированный обмен данными выполняется в течение 1 мс через каждые 4 мс работы непрерывной задачи.

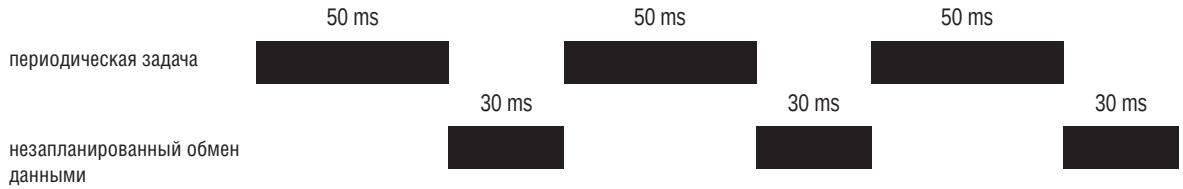


Если вы увеличите квант времени на служебные системные операции до 33%, то незапланированный обмен данными будет выполняться в течение 1 мс через каждые 2 мс работы непрерывной задачи.

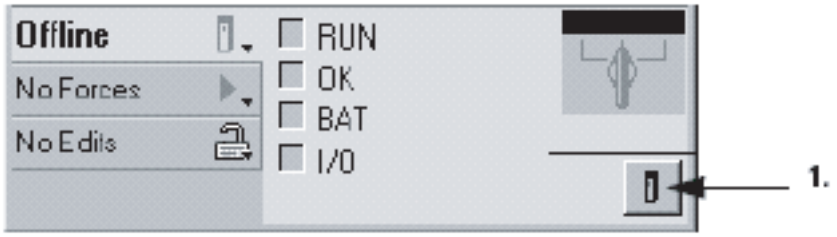


Если в контроллере находится только периодическая задача или несколько таких задач, то значение кванта времени на служебные системные операции не играет никакой роли. Незапланированный обмен данными выполняется тогда, когда не работает периодическая задача.

Например, если для выполнения вашей задаче требуется 50 мс, а вы установили периодичность ее обновления на 80 мс, то на незапланированный обмен данными у контроллера есть 30 мс из каждых 80 мс.

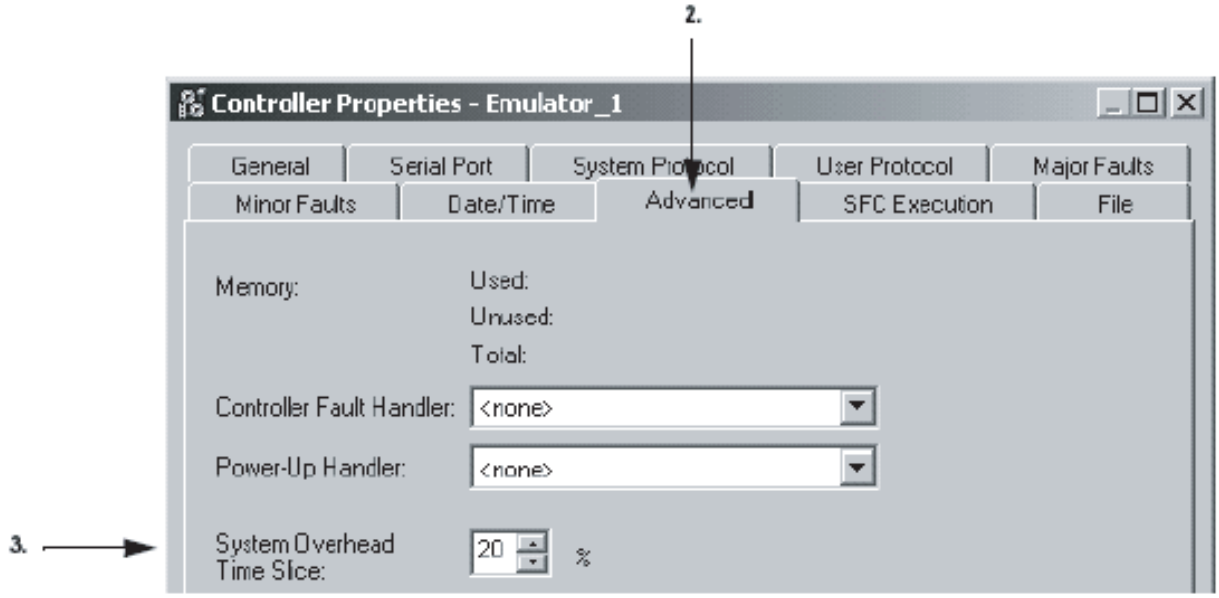


Корректировка кванта времени на служебные системные операции



1. На панели инструментов режима онлайн нажмите на кнопку свойств контроллера.

2. Щелкните по закладке *Advanced* (Дополнительные).



3. Введите или выберите квант времени на служебные системные операции.

4. Нажмите OK

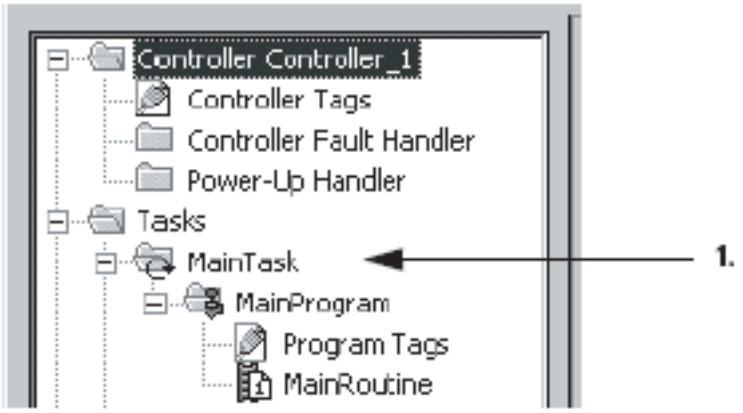
Просмотр времени сканирования

Контроллер Logix5000 обеспечивает два типа времени сканирования. Каждый из них имеет свое назначение:

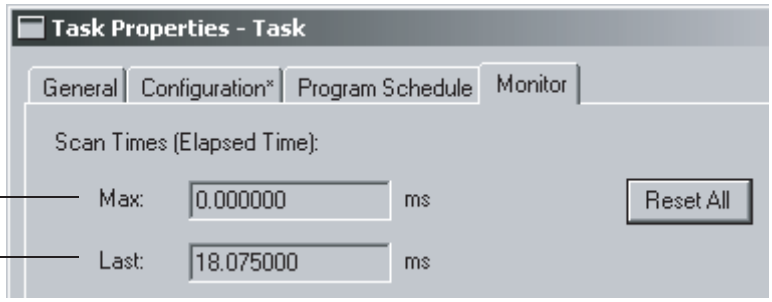
Если вы хотите определить:	То:	Примечания:
время в миллисекундах, прошедшее между началом и окончанием выполнения задача	Просмотрите время сканирования задачи	Время сканирования задачи включает время, в течение которого задача была прервана для служебного обмена данными или выполнения других задач.
время выполнения логики программы (ее главной процедуры и всех подпрограмм, вызываемых главной процедурой) в микросекундах	Просмотрите время сканирования программы	Время сканирования программы включает только время выполнения логики. Оно *не* включает никакие прерывания.

Просмотр времени сканирования задачи

Чтобы увидеть время сканирования задачи, выведите на экран свойства этой задачи.



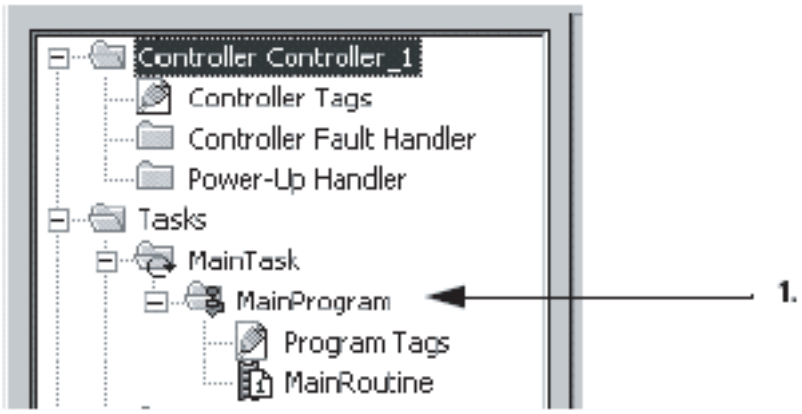
1. В организаторе контроллера щелкните правой кнопкой мыши по задаче, время сканирования которой вы хотите просмотреть, и выберите *Properties* (Свойства).
2. Щелкните по закладке *Monitor* (Монитор).



3. Чтобы закрыть это диалоговое окно, нажмит 

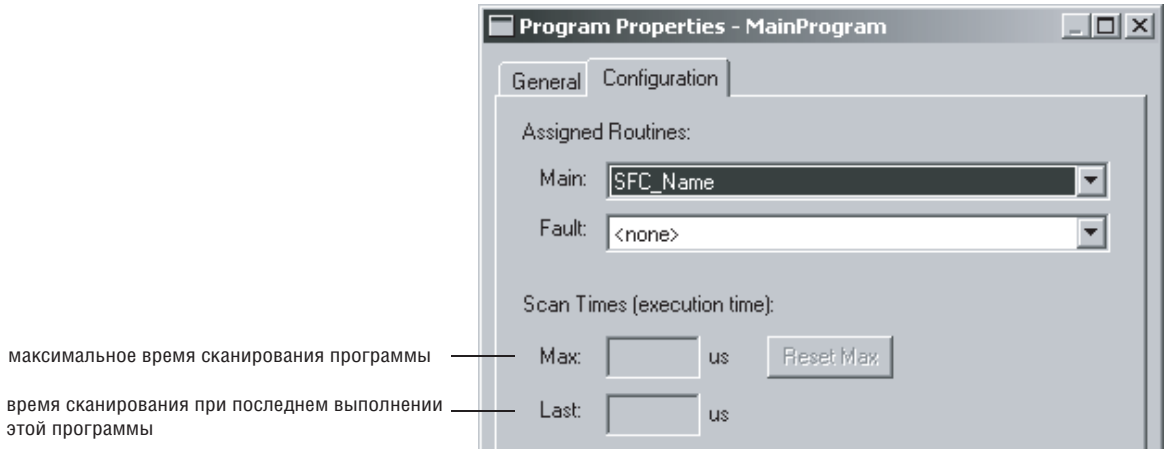
Просмотр времени сканирования программы

Чтобы увидеть время сканирования программы, выведите на экран свойства этой программы.



1. В организаторе контроллера щелкните правой кнопкой мыши по программе, время сканирования которой вы хотите просмотреть, и выберите *Properties* (Свойства).

2. Щелкните на закладке *Configuration* (Конфигурация).



3. Чтобы закрыть это диалоговое окно, нажмите OK.

Корректировка времени сторожевого таймера

В каждой задаче имеется сторожевой таймер, определяющий, через какое время работы задачи будет инициирована основная ошибка.

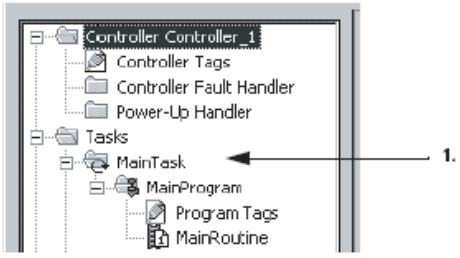
ВНИМАНИЕ

При достижении сторожевым таймером настраиваемого заданного времени возникает основная ошибка. В зависимости от обработчика ошибок контроллера может произойти останов контроллера.

- Время сторожевого таймера может варьироваться от 1 мс до 2000000 мс (2000 секунд). Значение по умолчанию – 500 мс.
- Сторожевой таймер начинает отсчет времени при запуске задачи и останавливается, когда будут выполнены все программы в данной задаче.
- Если время выполнения задачи превышает время сторожевого таймера, возникает основная ошибка. (Время включает прерывания другими задачами).
- Ошибка по превышению времени сторожевого таймера (основная ошибка) также возникает при повторном запуске задачи в процессе ее выполнения (наложение задач). Это может произойти в том случае, если низкоприоритетная задача прерывается высокоприоритетной, что задерживает выполнение низкоприоритетной задачи.
- Вы можете сбросить ошибку превышения контрольного времени при помощи обработчика ошибок контроллера. Если при сканировании той же самой логики вновь возникнет такая же ошибка превышения контрольного времени, контроллер перейдет в режим ошибки независимо от того, сбросил ли обработчик ошибок контроллера эту ошибку превышения контрольного времени.

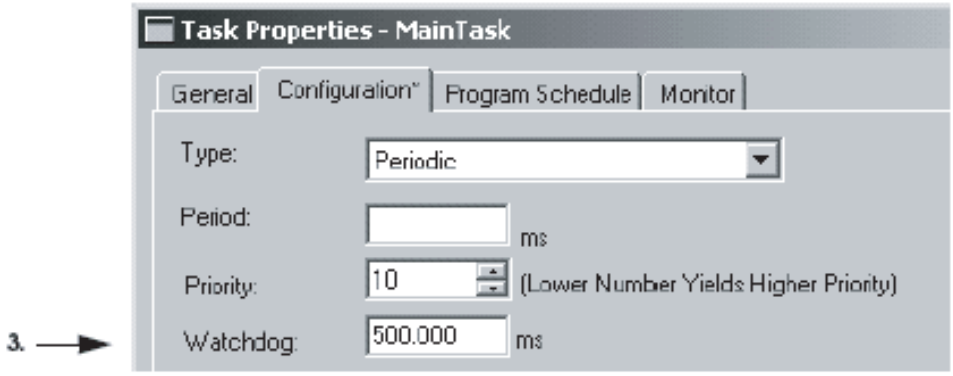
Корректировка времени сторожевого таймера для задачи

Чтобы изменить контрольное время для задачи, используйте диалоговое окно свойств этой задачи.



1. В организаторе контроллера щелкните правой кнопкой мыши по соответствующей задаче и выберите *Properties* (Свойства).

2. Щелкните по закладке *Configuration* (Конфигурация).



3. Введите время сторожевого таймера для этой задачи в миллисекундах.

4. Нажмите OK

Обмен данными с модулями ввода/вывода

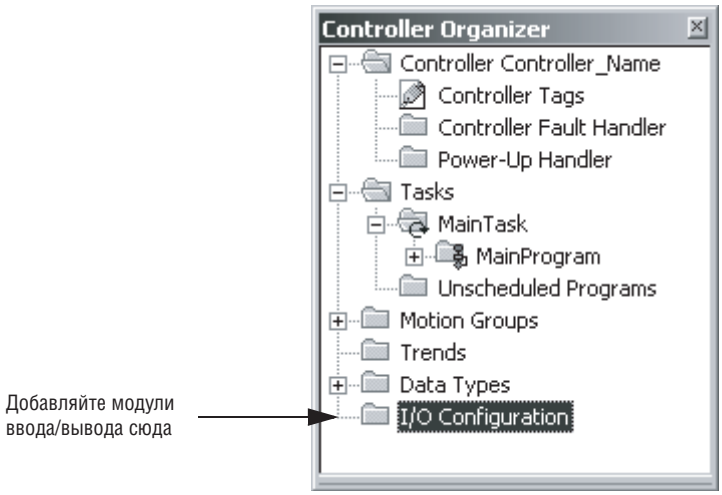
Использование этой главы

В этой главе содержится основополагающая информация по обмену данными между контроллером Logix5000 и модулями ввода/вывода.

За этой информацией или методикой	Обращайтесь к стр.:
Конфигурирование модуля ввода/вывода	2-1
Адресация данных ввода/вывода	2-7
Буферизация ввода/вывода	2-8

Конфигурирование модуля ввода/вывода

Чтобы обеспечить обмен данными с модулем ввода/вывода в вашей системе, добавьте соответствующий модуль в папку I/O Configuration (Конфигурация ввода/вывода) данного контроллера.



При добавлении модуля вы также задаете его конкретную конфигурацию. Хотя разные модули имеют разные конфигурируемые параметры, имеется и ряд общих опций, которые вам обычно требуется сконфигурировать:

- Requested Packet Interval (Запрашиваемый межпакетный интервал)
- Communication Format (Формат обмена данными)
- Electronic Keying (Электронный ключ)

Запрашиваемый межпакетный интервал

Контроллер Logix5000 использует соединения для передачи данных ввода/вывода.

Термин:	Определение:
Соединение	Линия связи между двумя устройствами, например, между контроллером и модулем ввода/вывода, терминалом PanelView или другим контроллером. Соединения создаются выделением ресурсов, обеспечивающим более надежную связь между устройствами, чем при передаче сообщений без соединения. Каждый контроллер может иметь ограниченное количество соединений. Вы косвенно определяете количество используемых контроллером соединений, конфигурируя обмен данными между данным контроллером и другими устройствами в вашей системе. Соединения используются следующими типами связи: • модули ввода/вывода • производимые и потребляемые теги • определенные типы инструкций Message (MSG) (не все типы этой инструкции используют соединение)
запрашиваемый межпакетный интервал (requested packet interval – RPI)	RPI определяет периодичность обновления данных посредством соединения. Например, модуль ввода направляет данные в контроллер с RPI, заданным вами для этого модуля. • Как правило, RPI задается в миллисекундах (мс). Допустимый диапазон – от 0,2 мс (200 микросекунд) до 750 мс. • Если устройства соединяются посредством сети ControlNet, то RPI резервирует интервал времени в потоке данных, проходящем по сети ControlNet. Время наступления этого интервала может не совпадать с точным значением RPI, но система управления гарантирует передачу данных как минимум с частотой, соответствующей RPI.

В контроллерах RSLogix5000 значения ввода/вывода обновляются с периодичностью, сконфигурированной вами посредством папки конфигурации ввода/вывода (I/O configuration) вашего проекта. Эти значения обновляются асинхронно по отношению к выполнению логики. Контроллер обновляет значение с заданной периодичностью, независимо от выполнения логики.

ВНИМАНИЕ

Позаботьтесь о том, чтобы в процессе выполнения задачи в памяти содержались адекватные значения. Вы можете скопировать данные или запомнить их в буфере в начале сканирования, чтобы обеспечить опорные значения для вашей логики.

- Программы задачи обращаются к данным ввода и вывода непосредственно в памяти, находящейся в области видимости контроллера.
- Логика любой задачи может изменять данные в области видимости контроллера.
- Данные и значения ввода/вывода являются асинхронными и могут изменяться в процессе выполнения задачи.
- Входное значение, к которому идет обращение в начале выполнения задачи, может быть другим при последующем обращении.
- Чтобы предотвратить изменение входного значения во время сканирования, скопируйте это значение в другой тег и используйте данные из него (запомните значения в буфере). Инструкции по буферизации ваших значений ввода/вывода см. на стр. 2-8.

Формат обмена данными

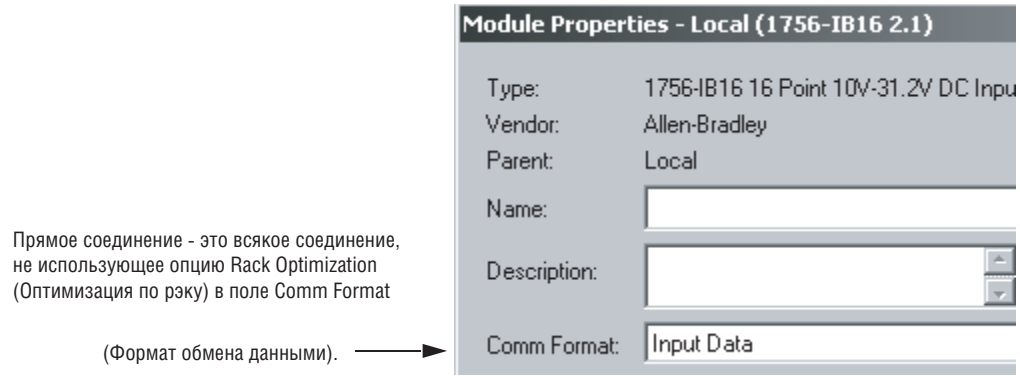
Выбранный вами формат обмена данными определяет структуру данных для тегов, связанных с определенными модулем. Многие модули ввода/вывода поддерживают различные форматы. Каждый формат использует свою структуру данных. Выбранный вами формат обмена данными также определяет:

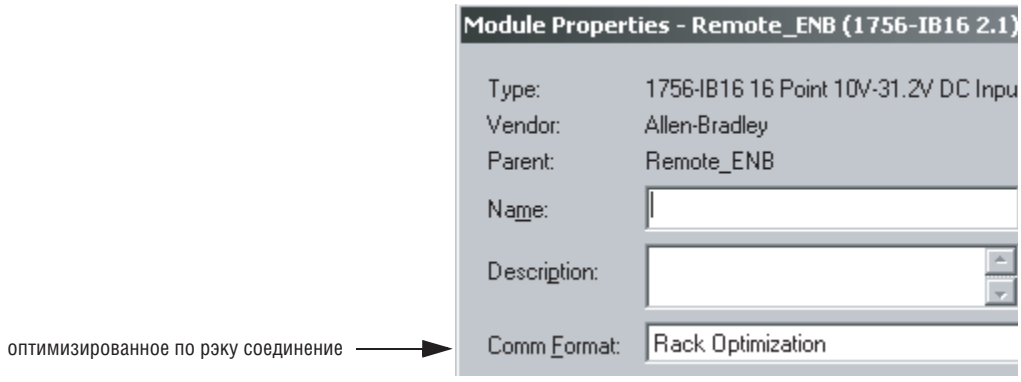
- Прямое или оптимизированное по рэку соединение
- Отношения собственности

Прямое или оптимизированное по рэку соединение

Контроллер Logix5000 использует соединения для передачи данных ввода/вывода. Эти соединения могут быть прямыми или оптимизированными по рэку.

Термин:	Определение:
прямое соединение	Прямое соединение – это линия передачи данных в реальном времени между контроллером и модулем ввода/вывода. Контроллер поддерживает и контролирует такое соединение с модулем ввода/вывода. Всякое нарушение соединения, например, ошибка модуля или удаление модуля при включенном питании, устанавливает биты ошибки в области данных, относящейся к соответствующему модулю.
оптимизированное по рэку соединение	Для модулей цифрового ввода/вывода вы можете выбрать оптимизированный по рэку обмен данными. Оптимизированное по рэку соединение консолидирует использование соединений между контроллером и модулями цифрового ввода/вывода в шасси (или на рейке стандарта DIN). Вместо отдельных прямых соединений с каждым модулем ввода/вывода имеется одно соединение для всего шасси (или рейки DIN).





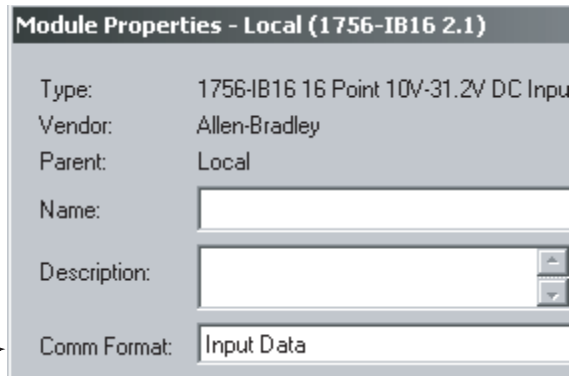
Отношения собственности

В системе Logix5000 модули осуществляют многоадресную передачу данных. Это означает, что несколько устройств могут одновременно получать одни и те же данные от одного устройства.

При выборе формата обмена данными вы должны задать, будет ли установлен собственник, или же контроллер по отношению к модулю будет лишь слушателем.

контроллер-собственник

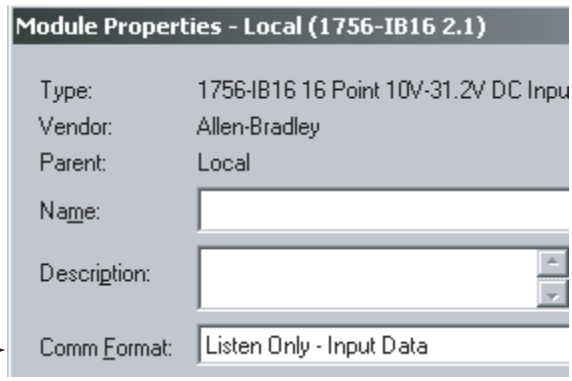
Контроллер, создающий основную конфигурацию и коммуникационное соединение с модулем. Контроллер-собственник записывает данные по конфигурации и может устанавливать соединение с модулем.



Соединение собственника - это любое соединение, в формате обмена данными которого (поле Comm Format) не содержится Listen-Only (Только прослушивание).

соединение только для прослушивания

Соединение с модулем ввода/вывода, при котором другой контроллер владеет данными по конфигурации/предоставляет такие данные модулю ввода/вывода. Контроллер, использующий соединение только для прослушивания, лишь контролирует соответствующий модуль. Он не записывает данные по конфигурации, и может поддерживать соединение с модулем ввода/вывода лишь в то время, когда контроллер-собственник активно управляет данным модулем ввода/вывода.



соединение только для прослушивания

Для выбора типа отношений собственности для модуля используйте следующую таблицу:

Если модуль является:	А другой контроллер:	И вы хотите:	То используйте этот тип соединения:
модулем ввода	*не* является собственником модуля	→	собственник (т.е. *не* только прослушивание)
	является собственником модуля	поддерживать связь с модулем, если он потеряет связь с другим контроллером	собственник (т.е. *не* только прослушивание) Используйте такую же конфигурацию, как у другого контроллера-собственника.
		прекратить связь с модулем, если он потеряет связь с другим контроллером	только прослушивание
модулем вывода	*не* является собственником модуля	→	собственник (т.е. *не* только прослушивание)
	является собственником модуля	→	только прослушивание

Управление модулями ввода заметно отличается от управления модулями вывода.

При управлении:	Это отношение собственности:	Означает:
модулями ввода	собственник	Модуль ввода конфигурируется контроллером, который устанавливает соединение в качестве собственника. Этот конфигурирующий контроллер является первым контроллером, устанавливающим соединение собственника. Когда модуль ввода будет сконфигурирован (и контроллер станет его собственником), другие контроллеры смогут установить соединения собственника с этим модулем. Это позволяет дополнительным собственникам продолжать получать широковещательные данные при нарушении соединения первоначального контроллера-собственника с данным модулем. Все остальные дополнительные собственники должны иметь идентичные с первоначальным контроллером-собственником данные по конфигурации и формат обмена данными, в противном случае попытка установить соединение будет отвергнута.
	только прослушивание	Когда модуль ввода будет сконфигурирован (и контроллер станет его собственником), другие контроллеры смогут установить соединение с этим модулем только для прослушивания. Эти контроллеры смогут получать широковещательные данные, при том, что собственником модуля является другой контроллер. Если будут нарушены соединения всех контроллеров-собственников с данным модулем ввода, все контроллеры с соединениями только для прослушивания прекратят получать широковещательные данные.
модулями вывода	собственник	Модуль вывода конфигурируется контроллером, который устанавливает соединение в качестве собственника. Для модуля вывода разрешается только соединение собственника. Если другой контроллер попытается установить соединение собственника, такая попытка будет отвергнута.
	только прослушивание	Когда модуль вывода будет сконфигурирован (и контроллер станет его собственником), другие контроллеры смогут установить соединения с этим модулем только для прослушивания. Эти контроллеры смогут получать широковещательные данные, при том, что собственником модуля является другой контроллер. Если будет нарушено соединение контроллера-собственника с данным модулем ввода, все контроллеры с соединениями только для прослушивания прекратят получать широковещательные данные

Электронный ключ

ВНИМАНИЕ

Будьте осторожны при отключении функции электронного ключа. При неправильном использовании эта опция может привести к травмам или смерти персонала, материальному ущербу или экономическим потерям.

При конфигурировании модуля вы задаете номер слота для данного модуля. Однако существует возможность преднамеренной или случайной установки другого модуля в этот слот.

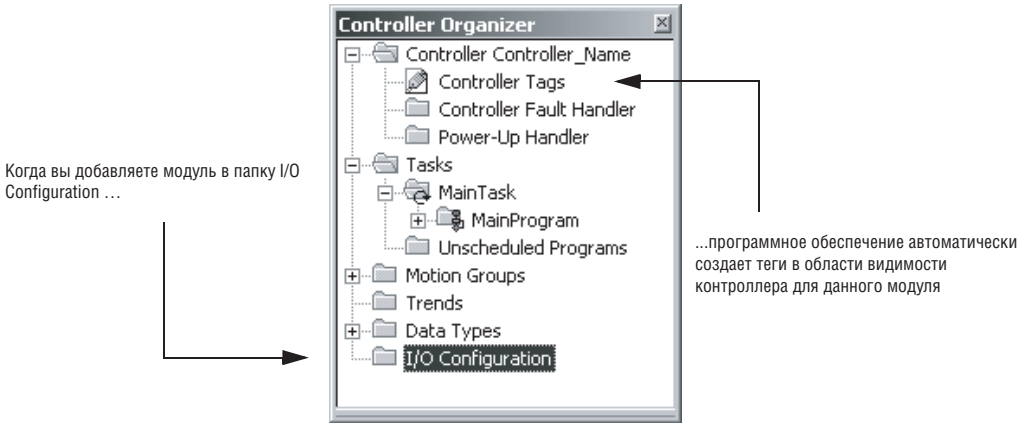
Электронный ключ позволяет вам защитить вашу систему от случайной установки в слот не того модуля. Задаваемая вами опция ключа определяет, насколько близко устанавливаемый в слот модуль должен соответствовать заданной для данного слота конфигурации.

Если:	**То выберите:**
вся информация должна соответствовать по: • типу • номеру по каталогу • поставщику • номеру основной и не основной ревизии	*Exact Match* (Точное соответствие)
соответствие всей информации, *кроме* номера не основной ревизии	*Compatible Module* (Совместимый модуль)
никакая информация не должна соответствовать	*Disable Keying* (Отключить ключ)

Адресация данных ввода/вывода

Информация ввода/вывода представляется в виде набора тегов.

- Каждый тег использует определенную структуру данных. Эта структура зависит от характеристик конкретного модуля ввода/вывода.
- Имена тегов определяются местоположением соответствующего модуля ввода/вывода в системе.



Адрес ввода/вывода имеет следующий формат:

Location | *:Slot* | *:Type* | *.Member* | *.SubMember* | *.Bit*

= Optional

Где:	Это:
Location	Местоположение в сети LOCAL = то же шасси или рейка DIN, что и у контроллера *ADAPTER_NAME* = обозначает удаленный коммуникационный адаптер или модуль моста
Slot	Номер слота модуля ввода/вывода в соответствующем шасси или рейке DIN
Type	Тип данных I = ввод O = вывод C = конфигурация S = статус
Member	Конкретные данные, поступающие от модуля ввода/вывода; зависят от того, данные какого типа могут храниться в данном модуле. • Для цифрового модуля, в члене Data (Данные) обычно хранятся двоичные значения ввода или вывода. • Для аналогового модуля, в члене Channel (CH#) (Канал) обычно хранятся данные для канала.
SubMember	Конкретные данные, относящиеся к члену (Member).
Bit	Конкретная точка в модуле цифрового ввода/вывода; зависит от размера модуля ввода/вывода (0-31 для 32-точечного модуля)

Буферизация ввода/вывода

В каких случаях использовать буферизацию ввода/вывода

Буферизация – это способ, при котором логика не обращается непосредственно к тегам реальных устройств ввода/вывода и не манипулирует ими. Вместо этого логика использует копию данных ввода/вывода. Используйте буферизацию ввода/вывода в следующих случаях:

- Для предотвращения изменения входного или выходного значения во время выполнения программы. (Ввод/вывод обновляется **асинхронно** по отношению к выполнению логики.)
- Для копирования тега ввода или вывода в член структуры или элемент массива.

Буферизация ввода/вывода

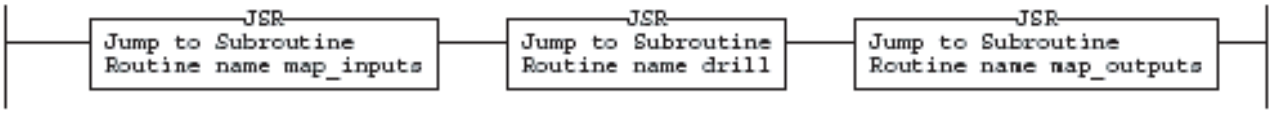
Для буферизации ввода/вывода выполните следующие действия:

1. В цепи перед логикой для функции (ий) скопируйте или переместите данные из требуемых тегов ввода в соответствующие им теги буфера.
2. В логике для функции (ий) обратитесь к тегам буфера.
3. В цепи после функции (ий) скопируйте данные из тегов буфера в соответствующие теги вывода.

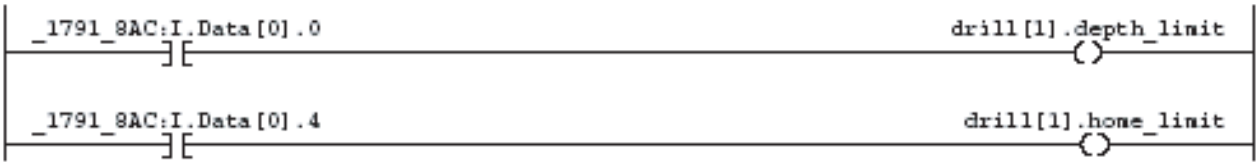
В следующем примере входные и выходные значения копируются в теги структуры для сверлильного станка.

ПРИМЕР Буферизация ввода/вывода

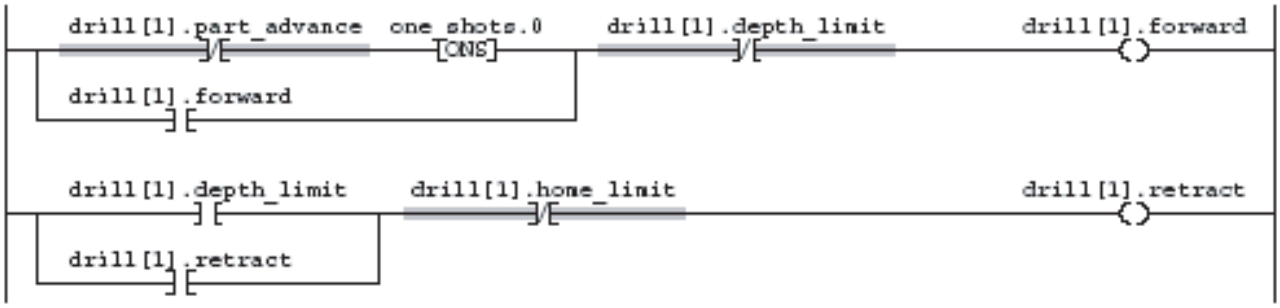
Главная процедура программы выполняет следующие подпрограммы в указанной последовательности.



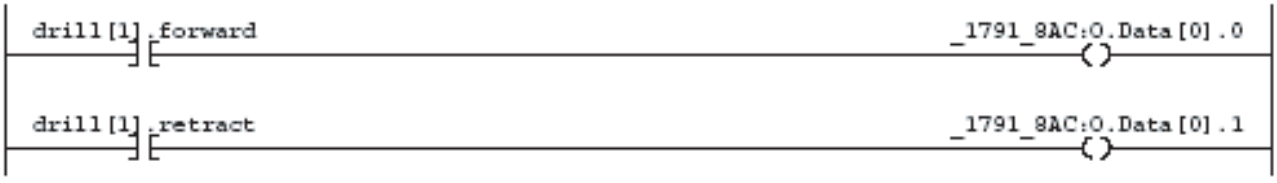
Процедура *map_inputs* копирует значения из устройств ввода в соответствующие теги, используемые в процедуре *drill*.



Процедура *drill* выполняет логику для сверлильного станка.



Процедура *map_outputs* копирует значения тегов вывода из процедуры *drill* в соответствующие устройства вывода.



В следующем примере используется инструкция CPS для копирования массива данных, отображающих устройства ввода в сети DeviceNet.

ПРИМЕР Буферизация ввода/вывода

В *Local:0:IData* хранятся входные данные для сети DeviceNet, подключенной к модулю 1756-DNB в слоте 0. Чтобы синхронизировать входы с приложением, инструкция CPS копирует входные данные в *input_buffer*.

- Во время копирования данных при помощи инструкции CPS никакие обновления ввода/вывода не могут изменить данные.
- При выполнении приложения оно в качестве своих входов использует входные данные, находящиеся в *input_buffer*.

Организация тегов

Использование этой главы

Используйте эту главу для организации данных для вашего контроллера Logix5000.

Определение тегов

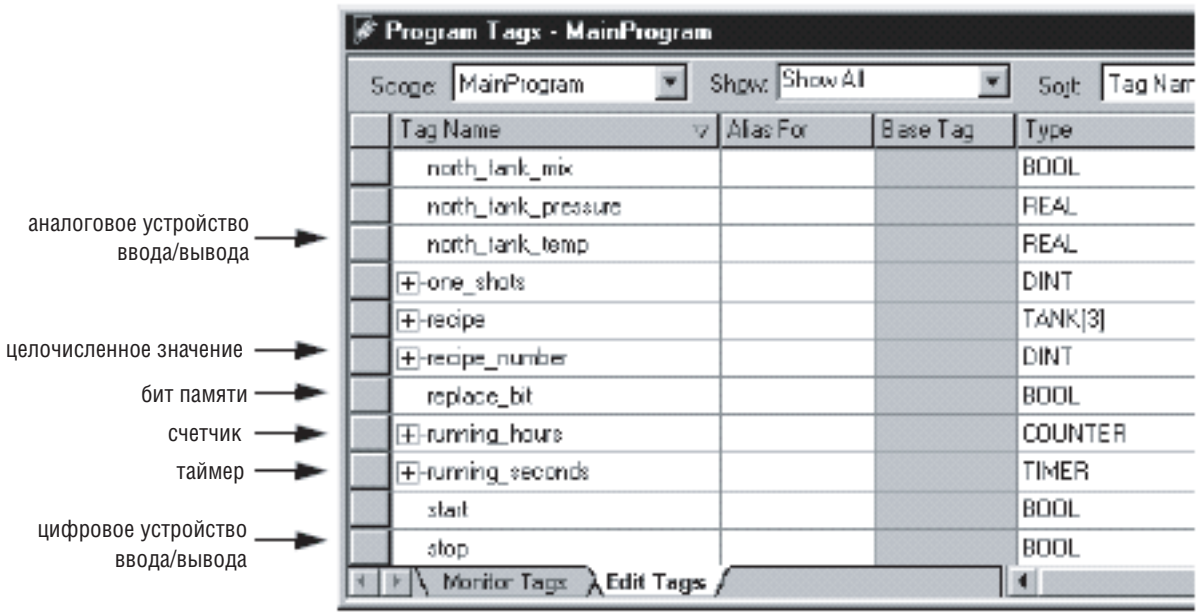
Для обращения к данным (параметрам) в контроллере Logix5000 используется тег (буквенно-цифровое имя).

Термин:	Определение:
тег	Текстовое имя области памяти контроллера, где хранятся данные. • Теги являются основным механизмом распределения памяти, обращения к данным из логики и контроля данных. • Минимальный размер памяти, выделяемой для тега – 4 байта. • Когда вы создаете тег для хранения данных, занимающих менее 4 байтов, контроллер выделяет 4 байта, но данные заполняют лишь требуемый им объем.

Контроллер использует имя тега для внутренних целей, и ему не требуются перекрестные ссылки на физический адрес.

- В традиционных PLC каждый элемент данных идентифицируется физическим адресом.
  - Адреса имеют фиксированный числовой формат, определяемый типом данных, например, N7:8, P8:3.
  - Для облегчения интерпретации логики требуются символьные обозначения.
- В контроллерах Logix5000 нет фиксированного числового формата. Данные идентифицируются самим именем тега. Это позволяет вам:
  - организовывать ваши данные таким образом, чтобы они отображали ваше машинное оборудование
  - документировать (при помощи имен тегов) ваше приложение в процессе его разработки

ПРИМЕР Теги



При создании тега вы присваиваете ему следующие свойства:

- Тип тега (Tag Type)
- Тип данных (Data Type)
- Область видимости (Scope)

Тип тега

Тип тега определяет действие данного тега в вашем проекте.

Если вы хотите, чтобы тег:	Выберите этот тип тега:
хранил значение или значения для использования логикой внутри проекта	Base (Базовый)
представлял другой тег	Alias (Псевдоним)
отправлял данные в другой контроллер	Produced (Производимый)
получал данные от другого контроллера	Consumed (Потребляемый)

Если вы собираетесь использовать производимые или потребляемые теги, вы должны выполнить дополнительные действия при организации ваших тегов. См. раздел «Обмен данными с другими устройствами» на стр. 10-1.

Тип данных

Термин:	Определение:
тип данных	Тип данных определяет тип хранимых в теге данных, например, двоичное, целочисленное значение, значение с плавающей точкой, строка и т.д.
структура	Тип данных, представляющий собой комбинацию других типов данных. • Структура имеет свой формат, создающий уникальный тип данных, предназначенный для конкретных целей. • Каждый отдельный тип данных внутри структуры называется членом. • Члены, как и теги, имеют имя и тип данных. • Контроллер Logix5000 содержит набор стандартных структур (типов данных), предназначенных для использования конкретными инструкциями, такими как таймеры, счетчики, функциональный блоки и т.д. • Вы можете создать свои собственные структуры, которые называются задаваемым пользователем (пользовательским) типом данных.

В следующей таблице указываются наиболее распространенные типы данных и их применение.

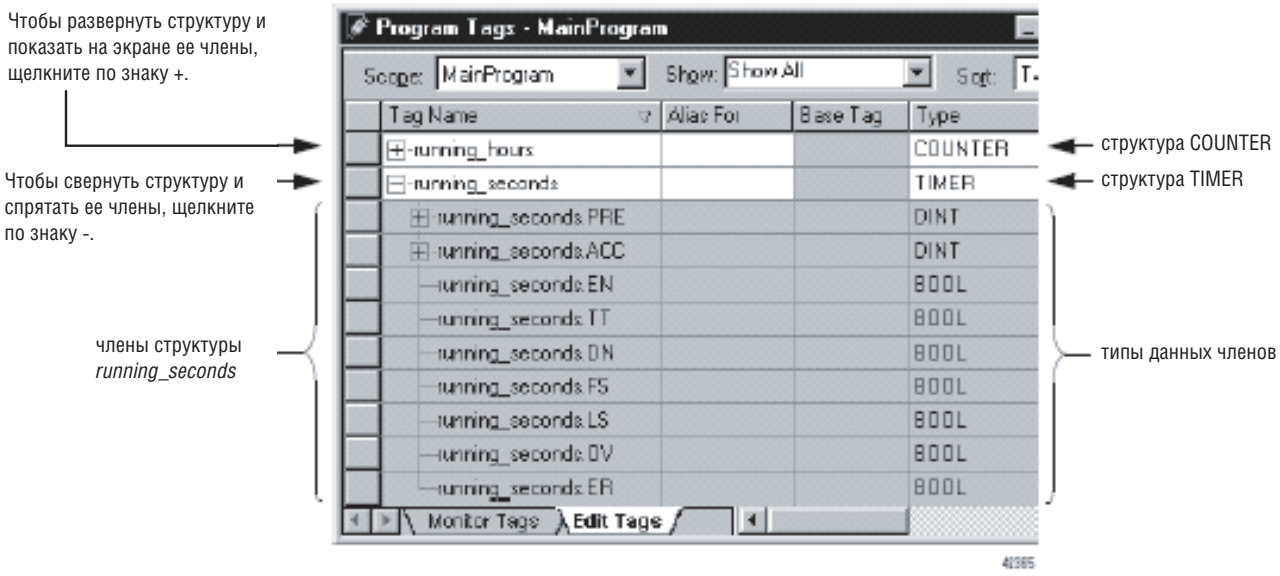
Таблица 3.1 Типы данных

Для:	Выберите:
аналогового устройства в режиме с плавающей точкой	REAL
аналогового устройства в целочисленном режиме (для очень большой частоты дискретизации)	INT
символов ASCII	string
бита	BOOL
счетчика	COUNTER
точки цифрового ввода/вывода	BOOL
числа с плавающей точкой	REAL
целого числа	DINT
секвенсера	CONTROL
таймера	TIMER

Минимальный размер памяти, выделяемый для тега – 4 байта. Когда вы создаете тег для хранения данных, занимающих менее 4 байтов, контроллер выделяет 4 байта, но данные заполняют лишь требуемый им объем.

Тип данных	Биты			
	31 16	**15 8**	**7 1**	**0**
Bool	не используется			0 or 1
Sint	не используется			-128 to +127
Int	не используется			-32,768 to +32767
Dint	-2,147,483,648 to +2,147,483,647			
Real	$-3.40282347E^{38}$ to $-1.17549435E^{-38}$ (negative values) 0 $1.17549435E^{-38}$ to $3.40282347E^{38}$ (positive values)			

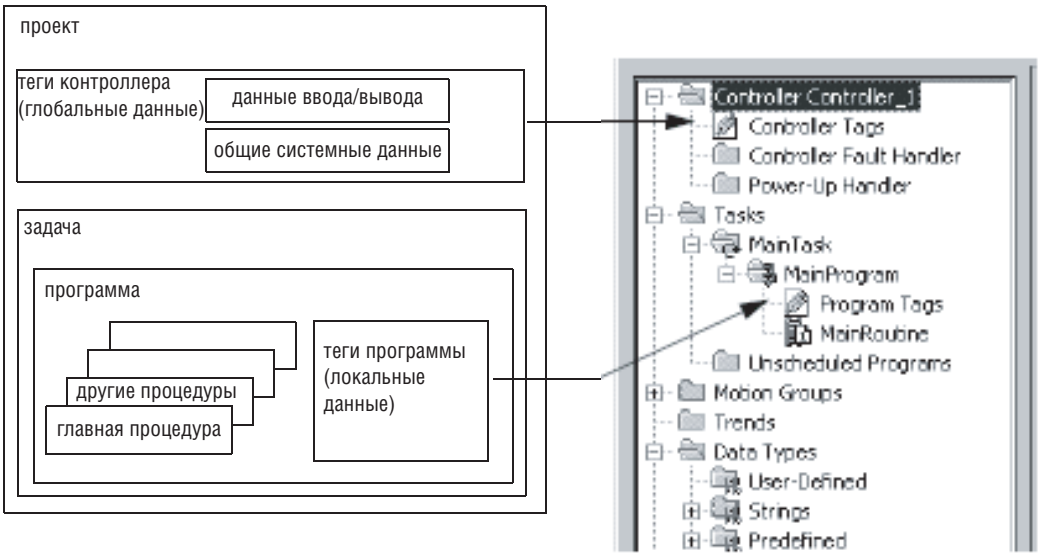
Типы данных COUNTER и TIMER являются примерами широко используемых структур.



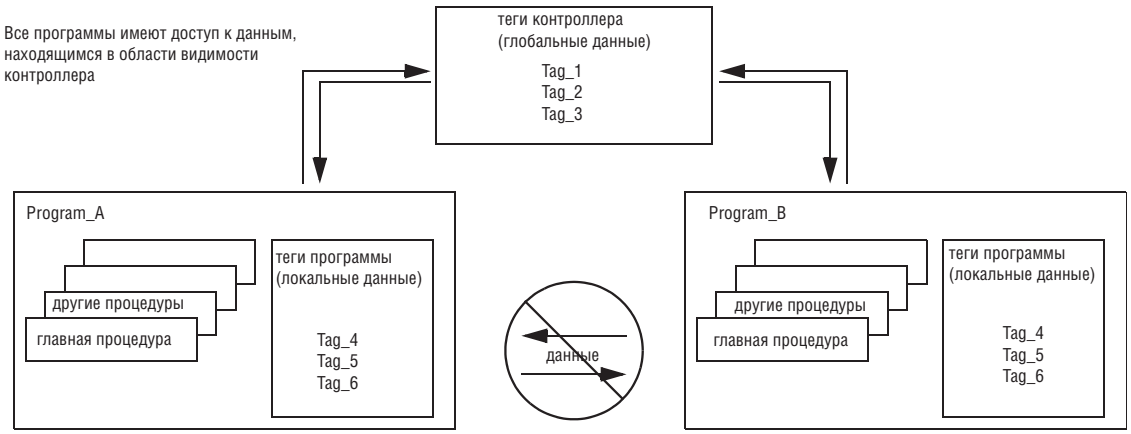
Чтобы скопировать данные в структуру, используйте инструкцию COP. Обратитесь к публикации 1756-RM003 «*Справочное руководство по общему набору инструкций контроллеров Logix5000*».

Область видимости

При создании тега вы определяете его либо как тег контроллера (глобальные данные), либо как программный тег для конкретной программы (локальные данные).



Контроллер Logix5000 позволяет вам разделить ваше приложение на несколько программ, со своими данными каждая. При этом *не* требуется управлять конфликтующими именами тегов из разных программ. Это облегчает многократное использование имен кодов и тегов в нескольких программах.



Данные в области видимости программы изолированы от других программ:

- Процедуры не могут обращаться к данным, находящимся в области видимости другой программы.
- Вы можете многократно использовать в нескольких программах имя тега, находящегося в области видимости программы. Например, обе программы (Program_A и Program_B) могут иметь тег программы с именем Tag_4.

Избегайте использования одного и того же имени для тега контроллера и тега программы. Внутри программы вы можете обращаться к тегу контроллера, если тег с таким же именем существует в качестве программного тега для данной программы.

Определенные теги должны находиться в области видимости контроллера (теги контроллера).

Если вы хотите использовать тег:	То задайте эту область видимости:
в нескольких программах проекта	область видимости контроллера (controller scope) (теги контроллера)
в инструкции Message (MSG)	
для производства или потребления данных	
для обмена данными с терминалом PanelView	
для других целей, кроме вышеперечисленного	область видимости программы (program scope) (теги программы)

Руководящие указания по созданию тегов

При создании тегов для проекта Logix5000 придерживайтесь нижеприведенных руководящих указаний:

Указание:	Подробное описание:
❑ 1. Создайте пользовательские типы данных – User Defined Datatype(UDT).	Пользовательские типы данных (структуры) позволяют вам организовать ваши данные таким образом, чтобы они соответствовали вашей установке или процессу. Пользовательский тип данных имеет следующие преимущества: • Один тег содержит все данные по определенному аспекту вашей системы. Это обеспечивает совместное размещение взаимосвязанных данных и простоту их поиска независимо от типа данных. • Каждая отдельная часть данных (член) получает описательное имя. Это автоматически создает начальный уровень документирования вашей логики. • Вы можете использовать этот тип данных для создания множества тегов при том же макете данных. Например, используйте пользовательский тип данных для хранения всех параметров резервуара, включая температуры, давления, позиции клапанов и предварительно установленные значения. Затем создайте тег для каждого из ваших резервуаров на основе этого типа данных.
❑ 2. Используйте массив для быстрого создания группы аналогичных тегов.	Массив создает множество экземпляров данных определенного типа под общим именем тега. • Массивы позволяют вам организовать блок тегов, использующих один и тот же тип данных и выполняющих аналогичную функцию. • Организуйте данные в одно-, двух- или трехмерные массивы в зависимости от того, что отображают эти данные. Например, используйте двухмерный массив для организации данных по резервуарному парку. Каждый элемент массива соответствует одному резервуару. Местоположение элемента в массиве соответствует географическому расположению резервуара. **Важно:** Минимизируйте использование массивов BOOL. Многие инструкции массивов *не* работают для массивов BOOL. Это затрудняет инициализацию и очистку массива данных типа BOOL. • Используйте массив BOOL в основном для объектов битового уровня экрана PanelView. • Также можно использовать отдельные биты тега DINT или массив данных типа DINT.
❑ 3. Воспользуйтесь тегами в области видимости программы.	Если вы хотите, чтобы у вас было множество тегов с одним и тем же именем, задайте каждый тег как программный тег (тег в области видимости программы) для другой программы. Это позволяет вам многократно использовать в разных программах и логику, и имена тегов. Избегайте использования одного и того же имени для тегов контроллера и программных тегов. Внутри программы вы не можете обращаться к тегу контроллера, если существует программный тег с таким же именем для этой программы. Определенные теги должны иметь область видимости контроллера (тег контроллера).

Если вы хотите использовать тег:	Задайте эту область видимости:
в нескольких программах проекта	область видимости контроллера (controller scope) (теги контроллера)
в инструкции Message (MSG)	
для производства или потребления данных	
для обмена данными с терминалом PanelView	
для других целей, кроме вышеперечисленного	область видимости программы (program scope) (теги программы)

Указание:	**Подробное описание:**
❑ 4. Для целочисленных значений используйте тип данных DINT.	Для увеличения эффективности вашей логики минимизируйте использование типов данных SINT и INT. Везде, где это возможно, для целочисленных значений используйте тип данных DINT. • Контроллер Logix5000 обычно выполняет сравнение или манипуляции со значениями как с 32-разрядными величинами (тип данных DINT или REAL). • Как правило, перед использованием значения контроллер преобразует значение SINT или INT в DINT или REAL. • Если приемником данных является тег SINT или INT, то контроллер обычно преобразует значение обратно в SINT или INT. • Преобразование в или из SINT или INT происходит автоматически, не требуя никакого дополнительного программирования. Однако оно требует дополнительного времени выполнения и дополнительной памяти.
❑ 5. Ограничьте имя тега 40 символами.	Правила для имени тега: • допускаются только буквы (A-Z или a-z), цифры (0-9) и знаки подчеркивания (_) • должно начинаться с буквы или знака подчеркивания • может включать не более 40 символов • не должно включать несколько знаков подчеркивания подряд или замыкающий знак подчеркивания (_) • *не* чувствительно к регистру
❑ 6. Используйте символы разных регистров	Хотя имена тегов не чувствительны к регистру (буква *A* верхнего регистра воспринимается так же, как буква *a* нижнего регистра), имена легче читаются при использовании символов разных регистров.

Эти теги легче читаются:	**Чем эти теги:**
Tank_1	TANK_1
Tank1	TANK1
	tank_1
	tank1

❑ 7. Учитывайте алфавитный порядок тегов.	Программное обеспечение RSLogix5000 показывает на экране теги одной и той же области видимости в алфавитном порядке. Для облегчения контроля взаимосвязанных тегов используйте одинаковые начальные символы для тегов, которые должны отображаться вместе.

Если каждый тег для резервуара будет начинаться с *Tank,* то эти теги будут отображаться вместе.

Tag Name
Tank_North
Tank_South
…

В противном случае эти теги могут оказаться далеко друг от друга.

Tag Name
North_Tank
…
…
…
South_Tank

← другие теги, начинающиеся с букв *o, p, q* и т.д.

Создание тега

ВАЖНО

RSLogix 5000 автоматически создает теги, когда вы:

- добавляете элемент в последовательную функциональную схему (ПФС)
- добавляете инструкцию функционального блока в функциональную блок-схему

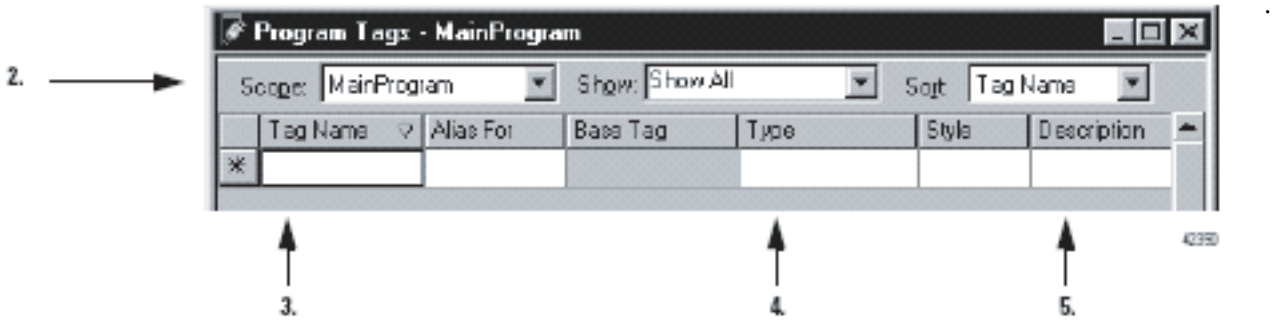
Вы можете создать тег несколькими способами:

- Создание тега при помощи окна Tags (Теги)
- Создание тега при помощи Microsoft® Excel
- Создание тега при ввода вашей логики (см. соответствующую главу для используемого вами языка программирования)

Создание тега при помощи окна Tags (Теги)

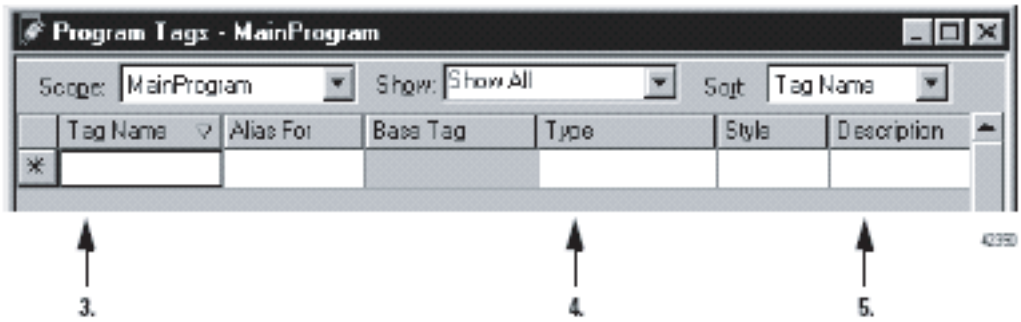
Окно Tags позволяет создавать и редактировать теги при помощи представления тегов в виде электронной таблицы.

1. В меню *Logic* (Логика) выберите *Edit Tags* (Редактирование тегов)



2.. Выберите **область видимости** для тега:

Если вы будете использовать тег:	То выберите:
в нескольких программах внутри проекта	*name_of_controller (контроллер)*
в качестве производителя или потребителя	
в сообщении	
только в одной программе внутри проекта	программу, которая будет использовать этот тег



3. Введите **имя** для данного тега.
4. Введите **тип данных**.
5. Введите **описание** (по желанию).

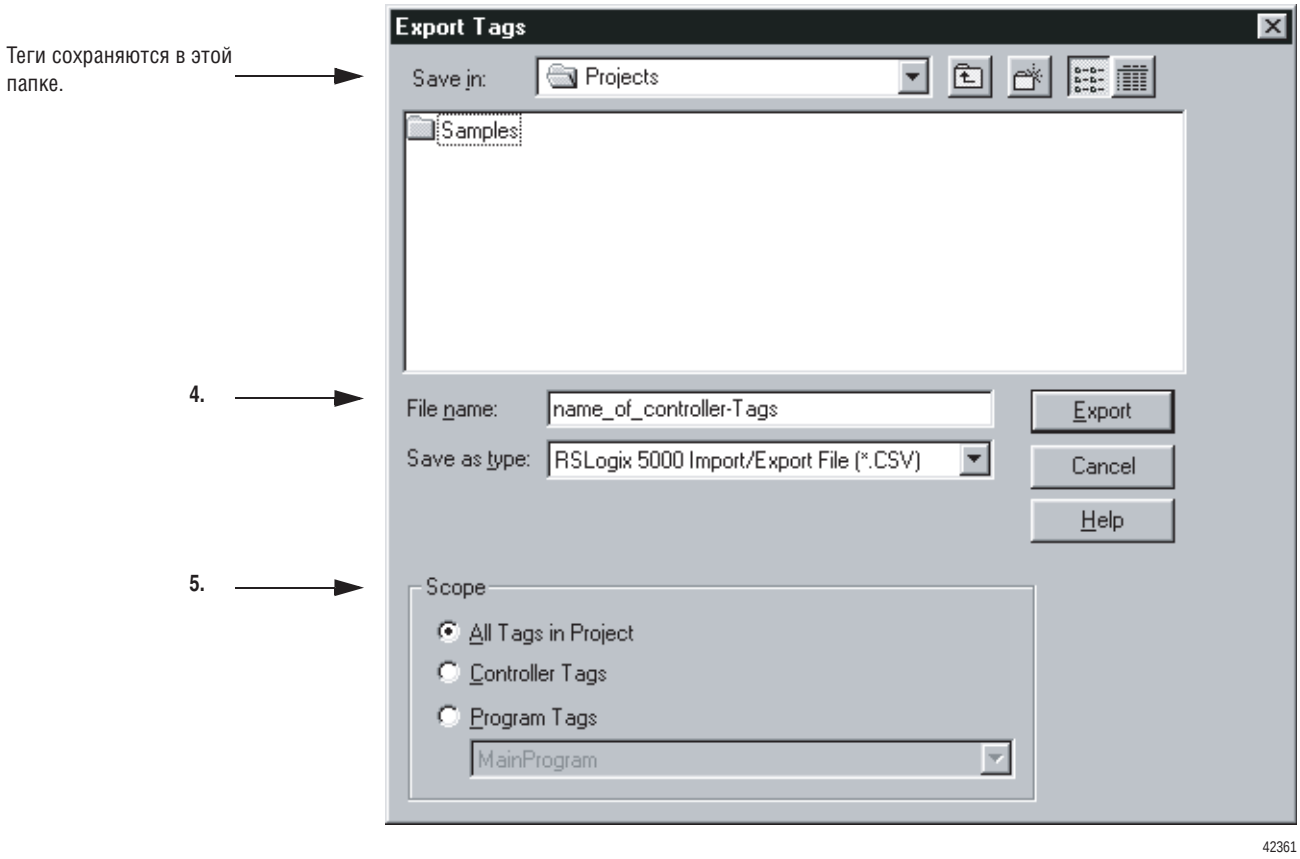
Создание тегов при помощи Microsoft® Excel

Вы также можете использовать программное обеспечение электронной таблицы типа Microsoft Excel для создания и редактирования тегов. Это позволяет вам воспользоваться возможностями редактирования, имеющимися в таком программном обеспечении.

Экспортируйте существующие теги

1. Откройте проект RSLogix 5000.
2. Создайте несколько тегов. (Это помогает отформатировать электронную таблицу Excel).

3. В меню *Tools* (Сервис) выберите *Export Tags* (Экспорт тегов).



4. Запомните имя файла экспорта (`project_name`-*Tags*).

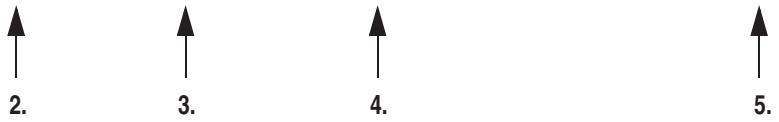
5. Выберите область видимости тегов, для которой должен производиться экспорт. Если вы выберите *Program Tags*, будут экспортироваться программные теги.

6. Нажмите *Export* (Экспортировать).

Отредактируйте файл экспорта

1. Откройте файл экспорта в Microsoft Excel.

TYPE	SCOPE	NAME	DESCRIPTION	DATATYPE
TAG		in_cycle		DINT
TYPE	SCOPE	NAME	DESCRIPTION	DATATYPE
TAG	MainProgram	conveyor_alarm		BOOL
TAG	MainProgram	conveyor_on		BOOL
TAG	MainProgram	drill_1		DRILL_STATION
TAG	MainProgram	hole_position		REAL[6,6]
TAG	MainProgram	machine_on		BOOL



2. Введите TAG

3. Укажите область видимости данного тега:

Если областью видимости является:	**То:**
контроллер	Оставьте эту ячейку пустой.
программа	Введите имя программы

4. Введите имя тега.

5. Введите тип данных для тега.

6. Повторите шаги со 2 по 5 для каждого дополнительного тега.

7. Сохраните и закройте файл. (Храните его в формате .CSV.)

Импортируйте новые теги

1. Выберите *Import Tags* (Импорт тегов) в меню *Tools* (Сервис) RSLogix 5000.

2. Выберите файл, содержащий теги, и нажмите *Import* (Импортировать).

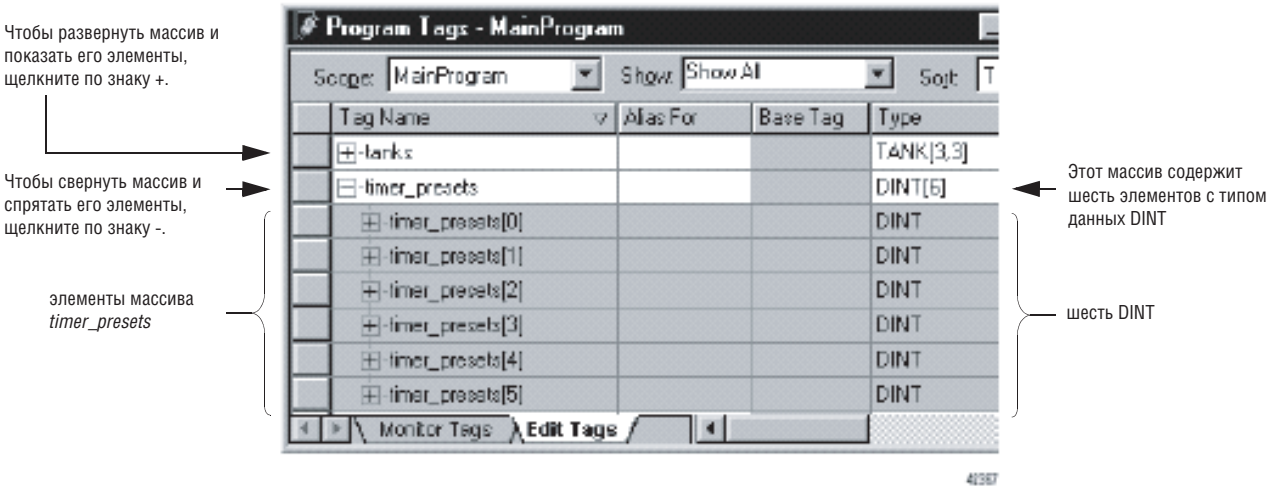
Теги будут импортированы в проект. В нижней части окна RSLogix 5000 вы увидите результаты этой операции.

Создание массива

Контроллеры Logix5000 также позволяют вам использовать массивы для организации данных.

Термин:	Определение:
массив	Тег, содержащий блок из многих элементов данных. • Массив аналогичен файлу. • Каждая отдельная часть данных внутри массива называется элементом. • Каждый элемент использует один и тот же тип данных. • Тег массива занимает в контроллере непрерывный блок памяти, один элемент за другим. • Вы можете использовать инструкции массива и секвенсера для манипулирования элементами массива или их индексации по всему массиву. • Вы можете организовать данные в одно-, дву- или трехмерный блок.

Индекс идентифицирует каждый отдельный **элемент** массива. Диапазон индексов – от 0 до количества элементов минус 1 (индексация с нуля).



В следующем примере структура сравнивается с массивом.

Это тег, использующий структуру (тип данных) Timer.

Tag Name	Data Type
Timer_1	TIMER
Timer_1.PRE	DINT
Timer_1.ACC	DINT
Timer_1.EN	BOOL
Timer_1.TT	BOOL
Timer_1.DN	BOOL

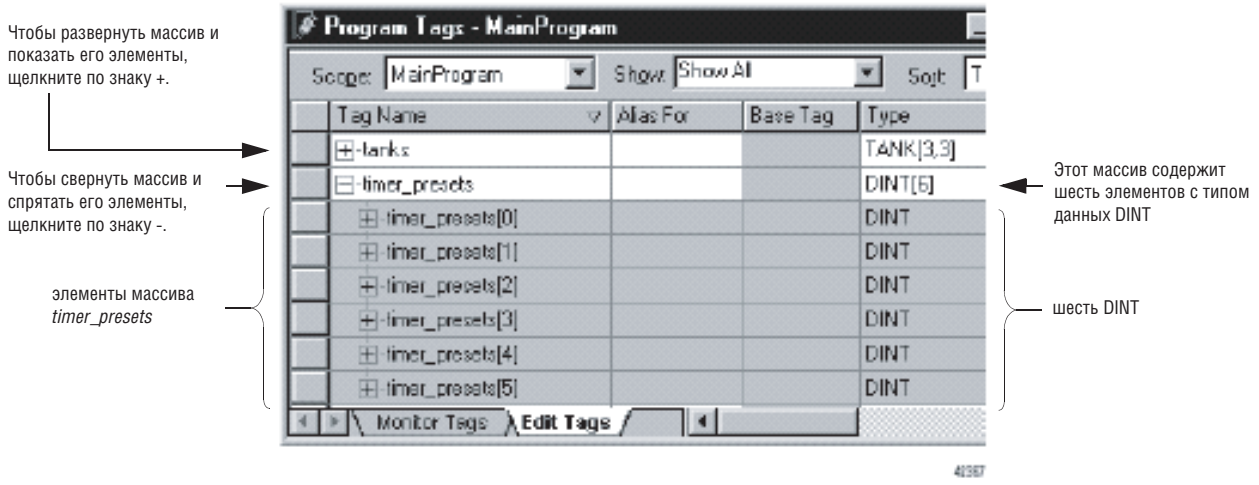
Это тег, использующий массив с типом данных Timer.

Tag Name	Data Type
Timers	TIMER[3]
Timer[0]	TIMER
Timer[1]	TIMER
Timer[2]	TIMER

ПРИМЕР

Одномерный массив

В этом примере одна инструкция таймера определяет продолжительность нескольких операций. Для каждой операции требуется свое предварительно заданное значение. Поскольку все значения имеют один и тот же тип данных (DINT), используется массив.



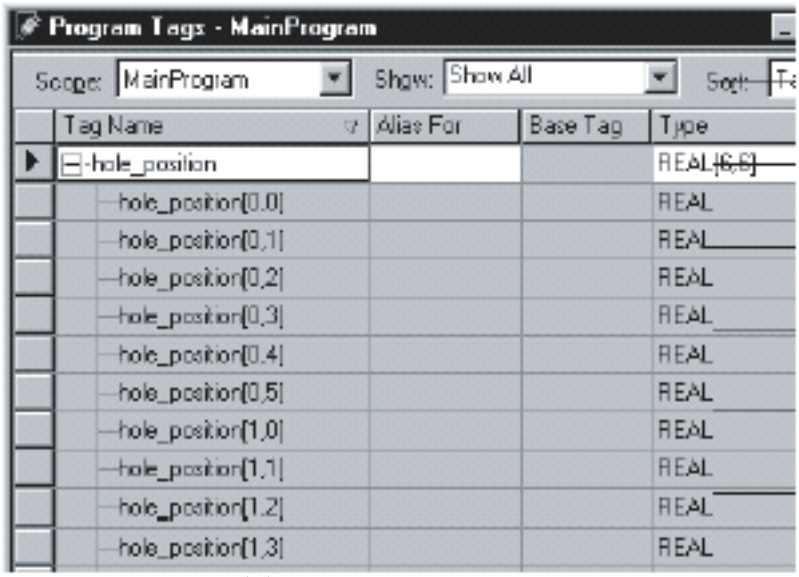
ПРИМЕР

Двумерный массив

Сверлильный станок может высверливать в книжном блоке от одного до пяти отверстий. Станку необходимо значение для положения каждого отверстия относительно передней кромки блока. Для организации этих значений в конфигурации используется двумерный массив. Первый индекс обозначает отверстие, которому соответствует данное значение, а второй указывает на то, сколько отверстий будет просверлено (от одного до пяти). .

индекс первой размерности	индекс второй размерности 0	1	2	3	4	5	Описание
0							
1		1.5	2.5	1.25	1.25	1.25	Расстояние первого отверстия от передней кромки блока
2			8.0	5.5	3.5	3.5	Расстояние второго отверстия от передней кромки блока
3				9.75	7.5	5.5	Расстояние третьего отверстия от передней кромки блока
4					9.75	7.5	Расстояние четвертого отверстия от передней кромки блока
5						9.75	Расстояние пятого отверстия от передней кромки блока

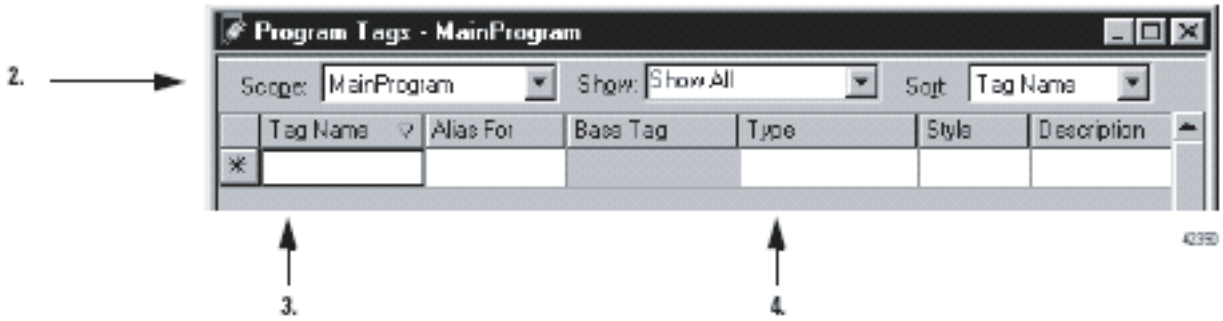
В окне Tags (Теги) элементы представлены в показанном ниже порядке.



Создание массива

Для создания массива вы должны создать тег и задать размерность для соответствующего типа данных.

1. В меню *Logic* (Логика) выберите *Edit Tags* (Редактирование тегов).



2. Выберите **область видимости** для данного тега:

Если вы будете использовать тег:	То выберите:
в нескольких программах внутри проекта	`name_of_controller` (контроллер)
в качестве производителя или потребителя	
в сообщении	
только в одной программе внутри проекта	программу, которая будет использовать этот тег

3. Введите **имя** тега.

4. Задайте размерность массива:

Если тег представляет собой:	То введите:	Где:
одномерный массив	*data_type [x]*	`data_type` – это тип данных, хранимых в теге.
двумерный массив	*data_type [x,y]*	`x` – это количество **элементов** первого измерения.
трехмерный массив	*data_type [x,y,z]*	`y` – это количество элементов второго измерения. `z` – это количество элементов третьего измерения.

Создание пользовательского типа данных (UDT)

Пользовательские типы данных (структуры) позволяют вам организовать ваши данные в соответствии с вашей установкой или процессом.

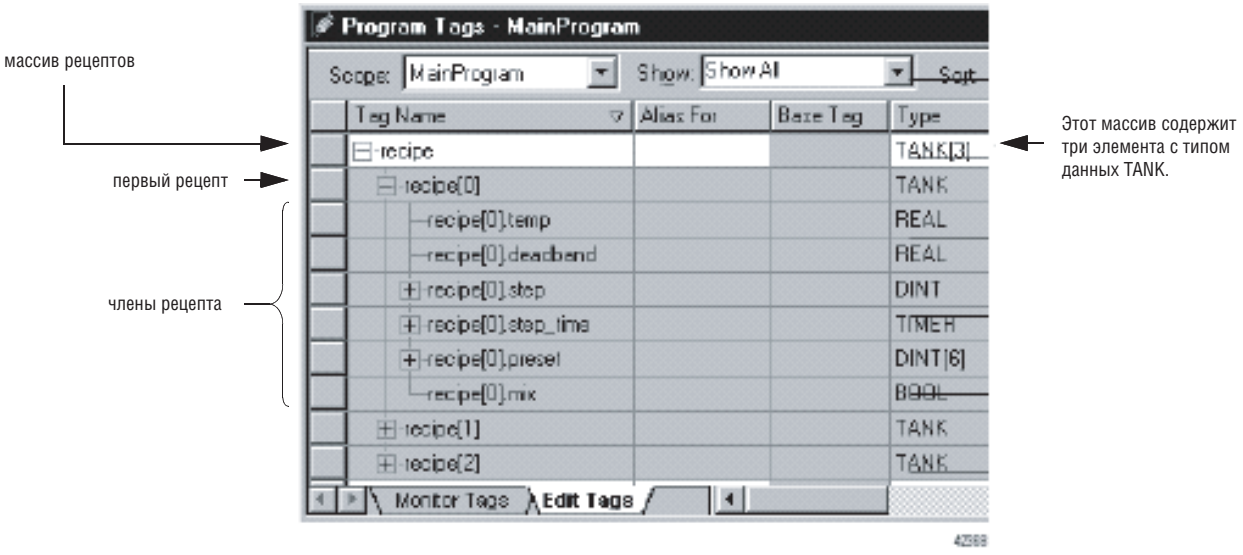
ПРИМЕР

Пользовательский тип данных для хранения рецепта

В системе из нескольких резервуаров для каждого из них могут использоваться разнообразные рецепты. Поскольку для рецепта требуется сочетание различных типов данных (REAL, DINT, BOOL и т.д.), используется пользовательский тип данных.

Имя (типа данных): TANK	
Имя члена	**Тип данных**
temp	REAL
deadband	REAL
step	DINT
step_time	TIMER
preset	DINT[6]
mix	BOOL

Массив, использующий такой тип данных, будет выглядеть следующим образом:



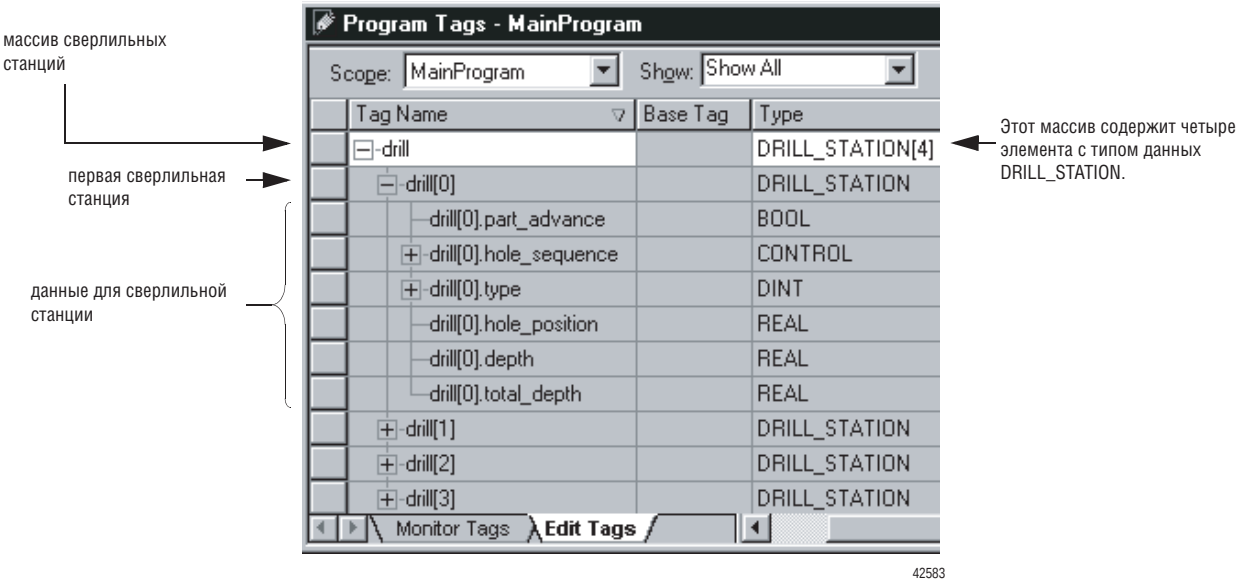
ПРИМЕР

Пользовательский тип данных для хранения данных, необходимых для работы станка

Поскольку для нескольких сверлильных станций требуется следующее сочетание данных, создается пользовательский тип данных.

Имя (типа данных): DRILL_STATION	
Имя члена	**Тип данных**
part_advance	BOOL
hole_sequence	CONTROL
type	DINT
hole_position	REAL
depth	REAL
total_depth	REAL

Массив, использующий такой тип данных, будет выглядеть следующим образом:



Руководящие указания по пользовательским типам данных (UDT)

При создании пользовательского типа данных помните о следующем:

- Если вы включаете члены для устройств ввода/вывода, вы должны использовать логику для копирования данных между членами структуры и соответствующими тегами ввода/вывода. См. раздел «Буферизация ввода/вывода» на стр. 2-8.
- Если вы включаете массив в качестве члена, ограничьте массив одним измерением. Многомерные массивы *не* допускаются в пользовательском типе данных.
- При использовании типов данных BOOL, SINT или INT размещайте члены одного типа данных друг за другом:

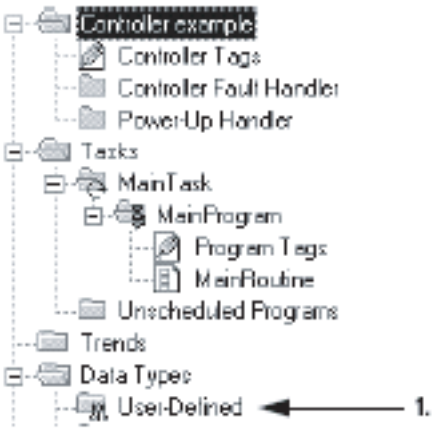
более эффективно

BOOL
BOOL
BOOL
DINT
DINT

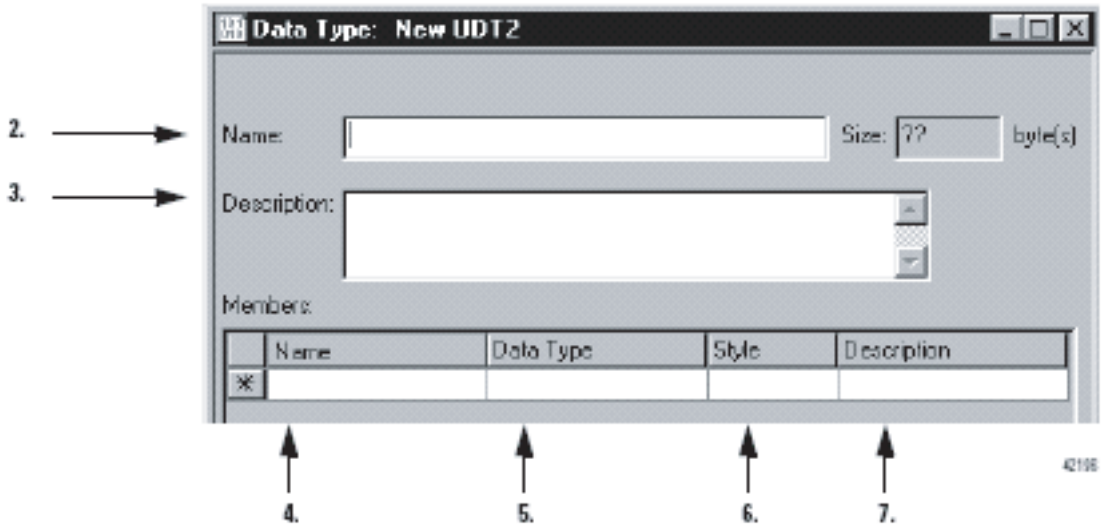
менее эффективно

BOOL
DINT
BOOL
DINT
BOOL

Создание пользовательского типа данных



1. Щелкните правой кнопкой мыши по *User-Defined* (Пользовательский) и выберите *New Data Type* (Новый тип данных).



2. Введите **имя** типа данных.

3. Введите **описание** (по желанию).

4. Введите имя первого **члена**.

5. Задайте тип данных для этого члена.
Ограничивайте массивы одним измерением.

6. Для отображения значения члена в другом **стиле** (другой системе счисления), выберите стиль.

7. Введите описание этого члена (по желанию).

8. Нажмите *Apply* (Применить).

9. Требуются дополнительные члены?

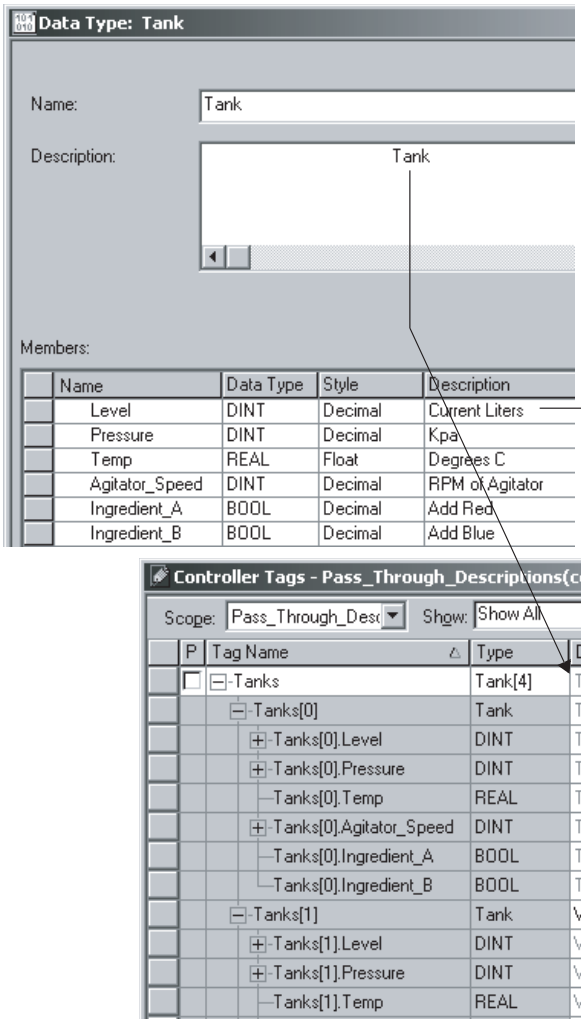
Если:	То:
Да	Повторите шаги с 4 по 8.
Нет	Нажмите *OK*.

Описание пользовательского типа данных (UDT)

RSLogix 5000 версии 13.0 и выше

Программное обеспечение RSLogix 5000 позволяет вам автоматически создавать описания на основе описаний, содержащихся в ваших пользовательских типах данных. Это значительно сокращает время, затрачиваемое вами на документирование вашего проекта.

При организации пользовательских типов данных помните о следующих возможностях программного обеспечения RSLogix 5000:



сквозное использование описаний - Когда это возможно, RSLogix 5000 ищет имеющееся описание для тега, элемента или члена:

- Описания в пользовательских типах данных переносятся на теги, использующие соответствующий тип данных.
- Описание тега массива переносится на элементы и члены этого массив

присоединение описания к базовому тегу - RSLogix 5000 автоматически формирует описание для каждого члена тега, использующего пользовательский тип данных. Оно начинается с описания данного тега, к которому присоединяется описание члена, взятое из типа данных.

вставка сквозного описания - Используйте описание типа данных и массива в качестве основы для более конкретных описаний. В данном примере Tank (Резервуар) превратился в West Tank (Западный резервуар).

Для описаний RSLogix 5000 использует различные цвета:

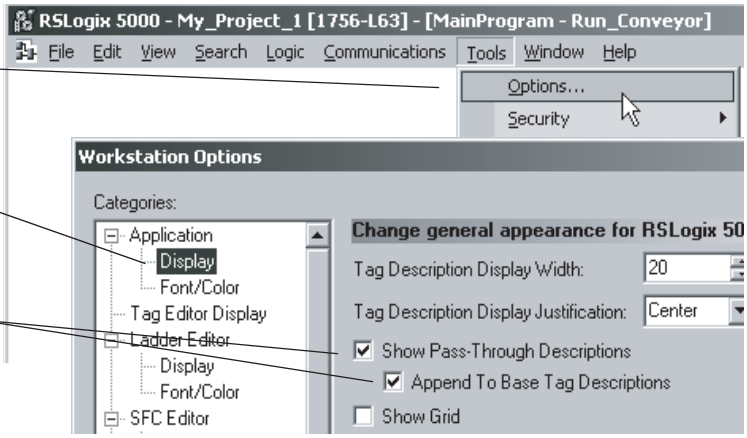
Описание этого цвета:	Соответствует:
серый	сквозному описанию
черный	описанию, введенному вручную

Включение и выключение сквозных (pass-through) и присоединяемых (append) описаний

1. В RSLogix 5000 выберите Tools (Сервис) ==> Options (Опции).

2. Выберите Application (Приложение) ==> Display (Отображение)

3. Включите (отметьте) или выключите (снимите отметку) соответствующие опции.

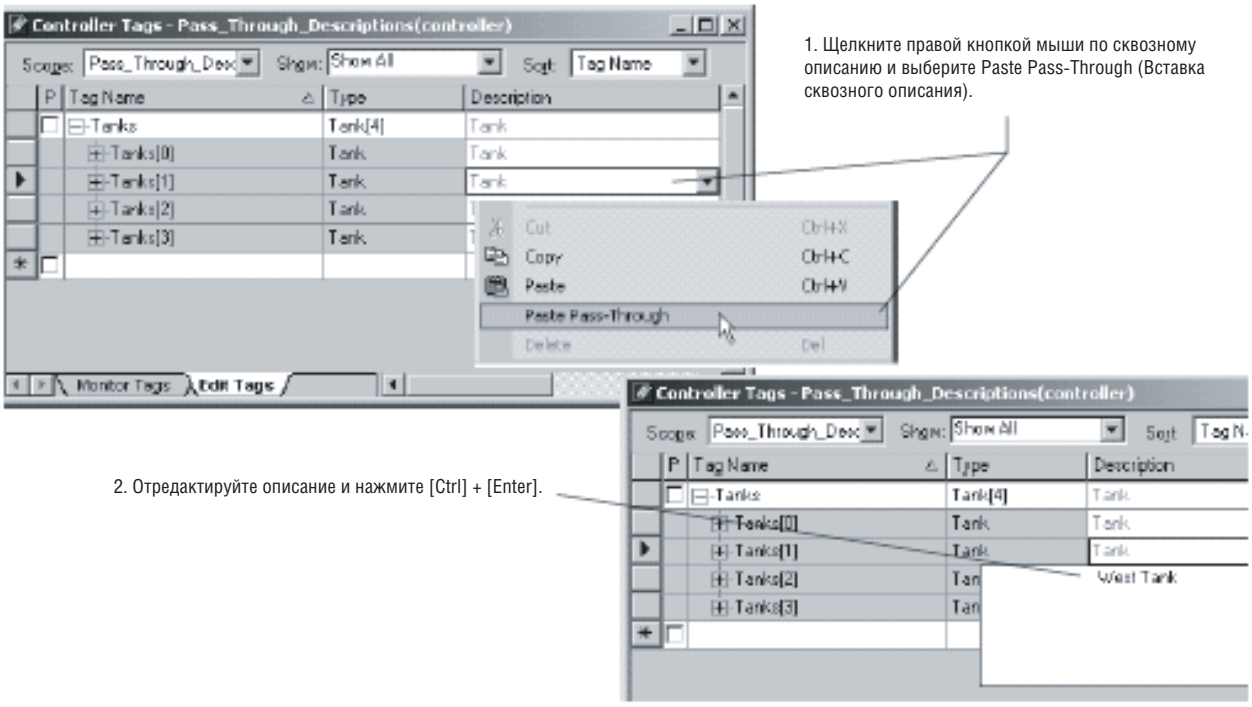


В RSLogix сквозных описаний

Для использования сквозного описания в качестве основы для более конкретного описания:

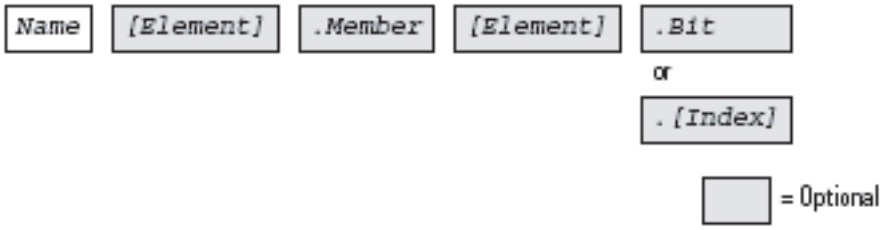
1. Щелкните правой кнопкой мыши по сквозному описанию и выберите Paste Pass-Through (Вставка сквозного описания).

2. Отредактируйте описание и нажмите [Ctrl] + [Enter].



Обращение к данным тега

Имя тега имеет следующий формат:



Где:	Это:
Name	Имя, идентифицирующее конкретный тег.
Element	Индекс или индексы, указывающие на конкретный элемент массива. • Используйте идентификатор элемента лишь в тех случаях, когда тег или член является массивом. • Используйте один индекс для каждого измерения массива. Например: [5], [2,8], [3,2,7]. Для косвенной (динамической) ссылки на элемент используйте тег или численное выражение, дающее номер элемента. • В численном выражении используется комбинация тегов, констант, операторов и функций для вычисления значения. Например, Tag_1 – Tag_2, Tag_3 + 4, ABS (Tag_4). • Значение тега или численного выражения должно находиться в пределах размерности массива. Например, если массив состоит из 10 элементов, то значение тега или численного выражения должно быть в диапазоне от 0 до 9 (10 элементов).
Member	Конкретный член структуры. • Используйте идентификатор члена лишь в тех случаях, когда тег является структурой. • Если структура содержит другую структуру в качестве одного из своих членов, используйте дополнительные уровни формата `.Member` для идентификации соответствующего члена.
Bit	Конкретный бит целочисленного типа данных (SINT, INT или DINT).
Index	Для косвенной (динамической) ссылки на бит целого числа используйте тег или численное выражение, дающее номер бита. • В численном выражении используется комбинация тегов, констант, операторов и функций для вычисления значения. Например, Tag_1 – Tag_2, Tag_3 + 4, ABS (Tag_4). • Значение тега или численного выражения должно находиться в пределах диапазона битов целочисленного тега. Например, если целочисленным тегом является Dint (32-bits), то значение индекса должно быть в пределах от 0 до 31 (32 бита).

Задание тегов-псевдонимов

Тег-псевдоним позволяет вам создать тег, который будет представлять другой тег.

- Оба тега используют одно и то же значение (значения).
- При изменении значения одного из таких тегов значение другого тега будет изменено таким же образом.

Используйте псевдонимы в следующих случаях:

- при программировании логики до выпуска монтажных схем
- для присвоения описательного имени устройству ввода/вывода
- для того, чтобы дать более простое имя сложному тегу
- для использования описательного имени для элемента массива

Информация о псевдонимах показана в окне тегов.

drill_1_depth_limit - это псевдоним для *Local:2:I.Data.3* (точки цифрового ввода). При включении ввода включается и тег-псевдоним.

drill_1_on - это псевдоним для *Local:0.O.Data.2 (*точки цифрового вывода). При включении тега-псевдонима включается и тег вывода.

north_tank - это псевдоним для *tanks[0,1]*.

Program Tags - MainProgram

Scope: MainProgram Show: Show All Sort: Tag Name

Tag Name	Alias For	Base Tag	Type
⊞-drill_1			DRILL_STAT
drill_1_depth_limit	Local:2:I.Data.3(C)	Local:2:I.Data.3(C)	BOOL
drill_1_forward	Local:0:O.Data.3(C)	Local:0:O.Data.3(C)	BOOL
drill_1_home_limit	Local:2:I.Data.2(C)	Local:2:I.Data.2(C)	BOOL
drill_1_on	Local:0:O.Data.2(C)	Local:0:O.Data.2(C)	BOOL
drill_1_retract	Local:0:O.Data.4(C)	Local:0:O.Data.4(C)	BOOL
⊞-hole_position			REAL[6,6]
machine_on			BOOL
⊞-north_tank	tanks[0,1]	tanks[0,1]	TANK
north_tank_drain			BOOL

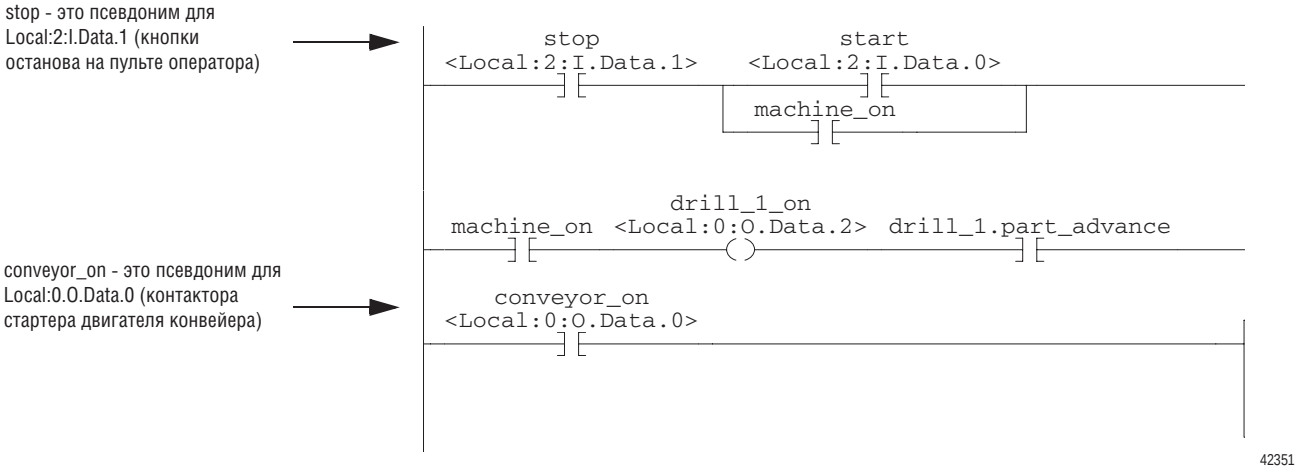
42360

(C) указывает на то, что данный тег находится в области видимости контроллера.

Как правило, теги-псевдонимы используются при программировании логики в отсутствие монтажных схем:

1. Для каждого устройства ввода/вывода создайте тег с именем, описывающим это устройство, например, *conveyor* для двигателя конвейера.
2. Запрограммируйте свою логику, используя описательные имена тегов. (Вы даже можете протестировать свою логику без подключения к устройствам ввода/вывода.)
3. Затем, когда будут готовы монтажные схемы, добавьте модули ввода/вывода в конфигурацию ввода/вывода данного контроллера.
4. Наконец, преобразуйте описательные теги в псевдонимы для соответствующих точек или каналов ввода/вывода.

Показанная ниже логика была первоначально запрограммирована с использованием описательных имен тегов, таких как *stop* и *conveyor_on*. Затем эти теги были преобразованы в псевдонимы для соответствующих устройств ввода/вывода.



Отображение информации о псевдонимах

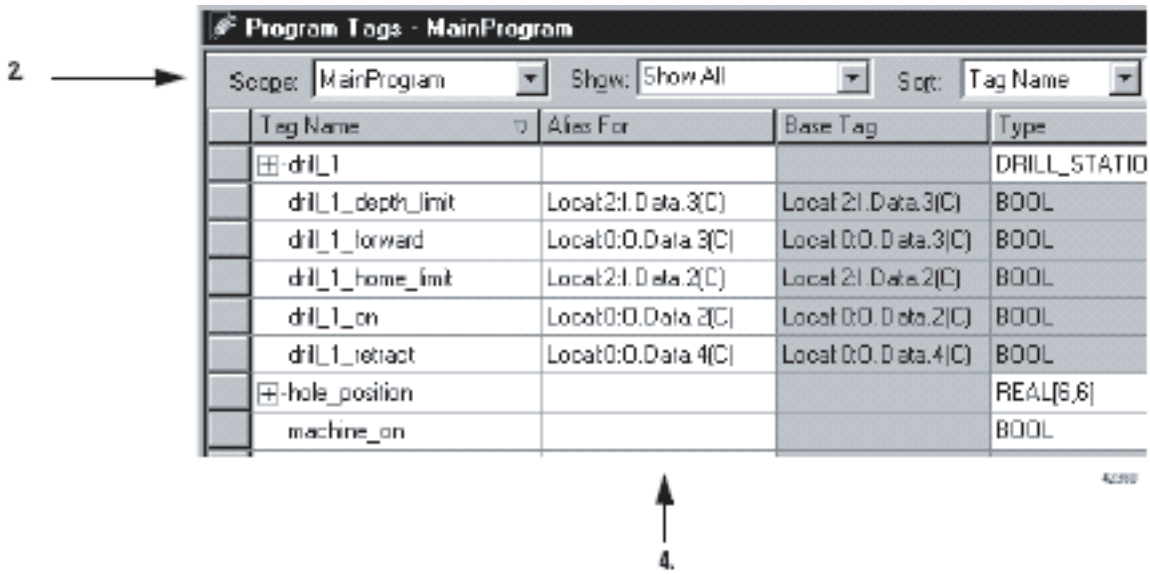
Чтобы показать (в вашей логике) тег, на который указывает псевдоним:

1. В меню *Tools* (Сервис) выберите *Options* (Опции).
2. Щелкните по закладке *Ladder Display* (Экран релейной логики).
3. Установите флажок *Show Tag Alias Information* (Показать информацию о псевдониме тега).
4. Нажмите ОК.

Задание псевдонима

Чтобы задать тег в качестве **тега-псевдонима** для другого тега:

1. В меню *Logic* (Логика) выберите *Edit Tags* (Редактирование тегов).



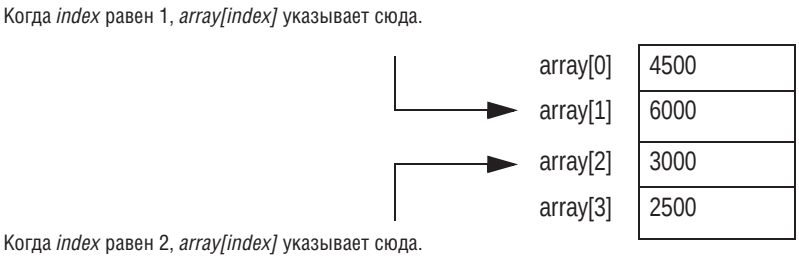
2. Выберите **область видимости** данного тега.
3. Щелкните по ячейке *Alias For* (Псевдоним для) справа от имени тега.
   В ячейке появится значок ▼
4. Щелкните по значку ▼
5. Выберите тег, которому будет соответствовать данный псевдоним:

Чтобы:	Выполните следующее:
выбрать тег	Дважды щелкните по имени тега.
выбрать номер бита	А. Щелкните по имени тега. Б. Щелкните по значку справа от имени тега. ▼ В. Щелкните по требуемому биту.

6. Нажмите на клавишу *Enter* или щелкните по другой ячейке.

Задание косвенного адреса

Если вы хотите, чтобы инструкция обращалась к различным элементам массива, используйте тег в индексе массива (косвенный адрес). Изменяя значение тега, вы изменяете элемент массива, к которому обращается ваша логика.



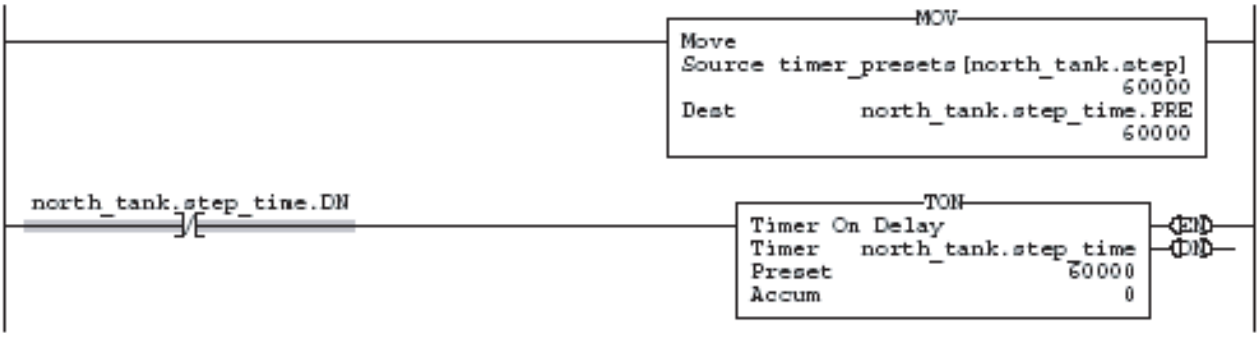
В следующей таблице указываются некоторые наиболее распространенные применения косвенного адреса:

Чтобы:	**Используйте тег в индексе и:**
выбрать рецепт из массива рецептов	Введите в тег номер соответствующего рецепта.
загрузить конкретную наладку станка из массива возможных наладок	Введите в тег нужную вам наладку.
поочередно, элемент за элементом, загрузить параметры или состояния из массива	А. Выполните необходимое действие над первым элементом. Б. Используйте инструкцию ADD для увеличения значения тега на единицу и перехода к следующему элементу массива.
занести в журнал регистрации коды ошибок	
выполнить несколько действий над элементом массива, а затем перейти к следующему элементу	

В следующем примере осуществляется загрузка ряда предварительно заданных значений в таймер, одно значение (элемент массива) за другим.

ПРИМЕР Перебор массива

В массиве *timer_presets* хранится ряд предварительно заданных значений для таймера в следующей цепочке. Тег *north_tank.step* указывает, какой элемент массива следует использовать. Например, когда *north_tank.step* равен 0, инструкция загружает в таймер *timer_presets[0]* (60000 мс).



После выполнения *north_tank.step_time* цепочка увеличивает значение *north_tank.step* до следующего номера, и в таймер загружается соответствующий элемент массива *timer_presets*.

Когда значение *north_tank.step* становится больше размера массива, цепочка возвращает данный тег к первому элементу массива. (Массив включает элементы с 0-го по 3-ий).

Выражения

Вы также можете использовать выражения для задания индекса массива.

- В выражении для вычисления значения используются операторы, такие как + или -.
- Контроллер вычисляет результат выражения и использует его в качестве индекса массива.

Для задания индекса массива вы можете использовать следующие операторы:

Оператор:	Описание:
+	сложение
-	вычитание/отрицание
*	умножение
/	деление
ABS	абсолютная величина
AND	И
FRD	перевод двоично-кодированного десятичного числа в целое

Оператор:	Описание:
MOD	остаток от деления
NOT	дополнение
OR	ИЛИ
SQR	корень квадратный
TOD	преобразование целого числа в двоично-кодированное десятичное
TRN	усечение
XOR	исключающее ИЛИ

Ваше выражение должно иметь следующий формат:

Если для оператора требуется:	Используйте этот формат:	Примеры:
одно значение (тег или выражение)	`operator(value)`	ABS(*tag_a*)
два значения (теги, константы или выражения)	`value_a operator value_b`	*tag_b* + 5 *tag_c* AND *tag_d* (*tag_e**2) MOD (*tag_f* / *tag_g*)

Управление многозадачным режимом

Использование этой главы

Проект RSLogix 5000, используемый по умолчанию, предлагает одну задачу для всей вашей логики. Хотя для многих приложений этого достаточно, в некоторых случаях может потребоваться несколько задач.

В этой главе содержится следующая информация по использованию в проекте нескольких задач:

Выбор задач контроллера

Контроллер Logix5000 позволяет вам использовать многозадачный режим для планирования выполнения ваших программ и назначения им приоритетов на основе определенных критериев. Это позволяет сбалансированно распределить время выполняемой контроллером обработки между различными операциями вашего приложения.

- В каждый момент времени контроллер выполняет только одну задачу.
- Другая задача может прервать выполняющуюся задачу и взять управление на себя.
- В каждый момент времени в каждой задаче выполняется только одна программа.

Контроллер Logix5000 использует три типа задач. Для выбора подходящего типа задачи для каждого участка вашей логики используйте следующую таблицу.

Если вы хотите, чтобы участок вашей логики выполнялся:	**То используйте этот тип задачи:**	**Описание:**
все время	Непрерывная задача (Continuous Task)	Непрерывная задача работает в фоновом режиме. Всякое время CPU, не выделенное для других операций (таких как перемещение, обмен данными и периодические или событийные задачи), используется для выполнения программ непрерывной задачи. • Непрерывная задача выполняется постоянно. Когда непрерывная задача завершает полный цикл сканирования, она немедленно перезапускается. • Для проекта не требуется непрерывная задача. Если такая задача используется, то она может быть только одна.
• с постоянной периодичностью (например, каждые 100 мс) • несколько раз в процессе сканирования другой вашей логики	Периодическая задача (Periodic Task)	Периодическая задача выполняет определенную функцию с заданной периодичностью. По истечении заданного для периодической задачи времени эта задача: • прерывает все задачи с более низким приоритетом • однократно выполняется • возвращает управление в место, где было прервано выполнение предшествующей задачи Вы можете задать период времени от 0.1 мс до 2000 с. • Значение по умолчанию – 10 мс. • Выполнение периодической задачи зависит от типа контроллера Logix5000 и от используемой в задаче логики.
сразу же после наступления какого-либо события	Событийная задача (Event Task)	Событийная задача выполняет определенную функцию только при наступлении заданного события (триггера). При возникновении триггера событийной задачи эта задача: • прерывает все задачи с более низким приоритетом • однократно выполняется • возвращает управление в место, где было прервано выполнение предшествующей задачи В качестве триггера может служить: • изменение цифрового входа • новая выборка аналоговых данных • определенные операции перемещения • потребляемый тег • инструкция EVENT **Важно:** Некоторые контроллеры Logix5000 поддерживают не все из этих триггеров. См. Таблицу 4.1 на стр. 4-21.

Ниже приводятся примеры возможных ситуаций и типы задач, которые можно было бы для них использовать:

В этом случае:	**Используйте этот тип задачи:**
Заполнение резервуара до максимального уровня с последующим открытием спускного клапана	непрерывная задача
Сбор и обработка параметров системы и их вывод на экран	непрерывная задача
Выполнение шага 3 в управляющей последовательности – перестановка отводящей перегородки бункера	непрерывная задача
Ваша система должна проверять положение механической руки каждые 0,1 с и вычислять среднюю скорость изменения его положения. Это используется для определения давления торможения.	периодическая задача
Считывание толщины рулона бумаги каждые 20 мс.	периодическая задача
На упаковочной линии производится склеивание коробок. При поступлении коробки на операцию склеивания контроллер должен немедленно выполнить процедуру склеивания.	событийная задача
При выполнении высокоскоростной сборки оптический детектор обнаруживает брак определенного типа. При обнаружении детектором брака он должен быть немедленно отброшен.	событийная задача
На испытательном стенде двигателя вы хотите получать и архивировать все аналоговые данные сразу после каждой выборки данных	событийная задача
Загрузка данных в станцию сразу после получения новых производственных данных	событийная задача
На линии упаковки конфет необходимо обеспечить перфорацию каждой конфеты в заданном месте. При каждом обнаружении датчиком регистрации соответствующей отметки следует проверить точность оси и при необходимости выполнить ее корректировку.	событийная задача
Станция склеивания должна корректировать количество наносимого клея для компенсации изменений скорости перемещения оси. После выполнения планировщика перемещения необходимо проверить заданную командой скорость перемещения оси и при необходимости изменить количество клея.	событийная задача
Вся производственная линия должна быть остановлена при обнаружении какой-либо программой опасного состояния. Процедура останова не зависит от типа опасного состояния.	событийная задача

Количество поддерживаемых задач зависит от контроллера:

Этот контроллер:	Поддерживает это количество задач:	Примечание:
ControlLogix SoftLogix5800	32	Только одна задача может быть непрерывной.
CompactLogix DriveLogix FlexLogix	8	

С осторожностью подходите к количеству используемых вами задач

Как правило, каждая задача забирает время контроллера у других задач. Если у вас слишком много задач, то:

- Непрерывная задача может выполняться слишком долго.
- Другие задачи могут перекрываться. Если задача прерывается слишком часто или на слишком длительное время, она может не успеть выполниться до ее очередного запуска.

За дополнительной информацией обращайтесь к разделу «Избежание перекрытий» на стр. 4-9.

Назначение приоритетов периодическим и событийным задачам

Хотя проект может включать несколько задач, в каждый данный момент времени контроллер выполняет лишь одну задачу. Если во время выполнения какой-либо задачи запускается периодическая или событийная задача, контроллер определяет, что ему делать, исходя из приоритета каждой из задач.

Количество уровней приоритета зависит от контроллера:

Этот контроллер Logix5000:	Имеет столько уровней приоритета:
CompactLogix	15
ControlLogix	15
DriveLogix	15
FlexLogix	15
SoftLogix5800	3

Для назначения задаче приоритета воспользуйтесь следующими указаниями:

Если вы хотите, чтобы:	То	Примечания:
данная задача прерывала другую задачу	Назначьте номер приоритета, меньший (более высокий приоритет), чем номер приоритета другой задачи.	• Задача с более высоким приоритетом прерывает все задачи, приоритет которых ниже. • Задача с более высоким приоритетом может многократно прерывать задачу, приоритет которой ниже.
другая задача прерывала данную задачу	Назначьте номер приоритета, больший (более низкий приоритет), чем номер приоритета другой задачи.	
данная задача разделяла время контроллера с другой задачей	Назначьте обеим задачам одинаковый номер приоритета.	Контроллер переключается между задачами и выполняет каждую из них в течение 1 мс.

Дополнительные факторы

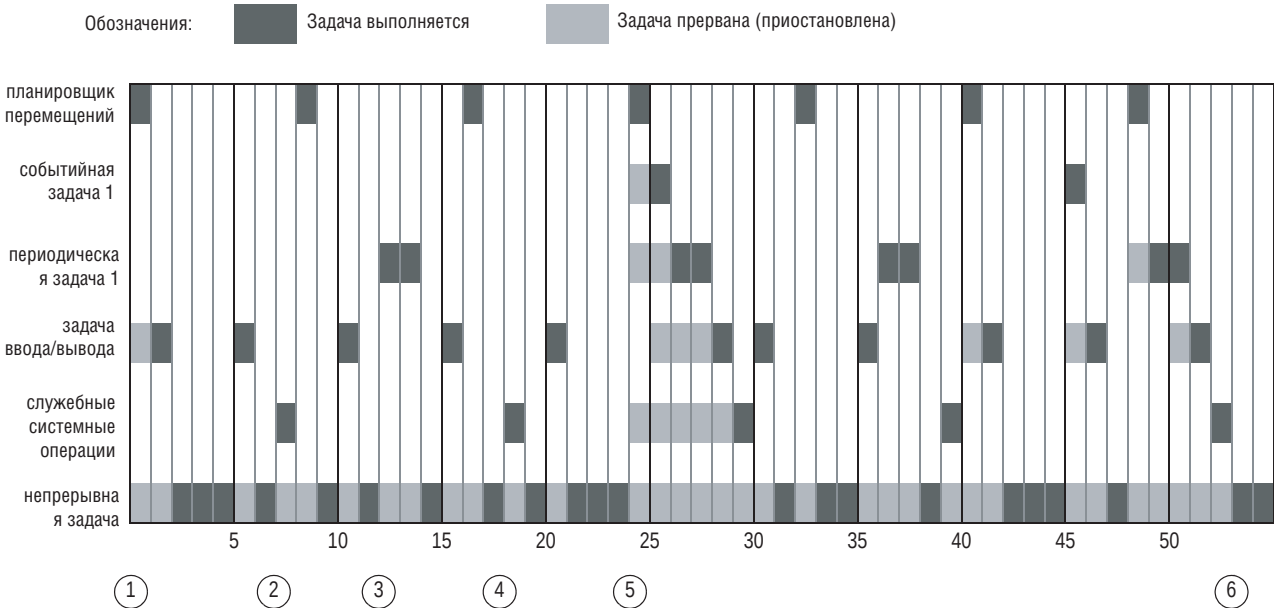
При оценке количества прерываний выполнения какой-либо задачи учитывайте следующие факторы:

<table>
<tr><th>Фактор:</th><th>Описание:</th></tr>
<tr><td>планировщик перемещений</td><td>Планировщик перемещений прерывает все остальные задачи независимо от их приоритетов.
Количество осей и периодичность грубого обновления для группы перемещения влияют на продолжительность и частоту выполнения планировщика перемещений.
Если планировщик перемещений выполняется во время запуска задачи, то задача будет ожидать завершения его выполнения.
Если наступает время грубого обновления во время выполнения задачи, то задача приостанавливается для выполнения планировщика перемещений.</td></tr>
<tr><td>задача ввода/вывода</td><td>Контроллеры CompactLogix, FlexLogix и DriveLogix используют специализированную периодическую задачу для обработки данных ввода/вывода. Эта задача ввода/вывода:
Не указывается в папке Tasks (Задачи) контроллера.
Не учитывается при подсчете допустимого количества задач для данного контроллера.
Работает с приоритетом 7.
Выполняется с максимальной частотой (RPI), запланированной вами для данной системы.
Выполняется столько времени, сколько требуется для сканирования сконфигурированных модулей ввода/вывода.
При назначении приоритетов своим задачам учитывайте задачу ввода/вывода:<table><tr><th>Если вы хотите, чтобы задача:</th><th>То назначьте один из этих приоритетов:</th></tr><tr><td>прерывала или задерживала обработку ввода/вывода</td><td>от 1 до 6</td></tr><tr><td>разделяла время процессора с обработкой ввода/вывода</td><td>7</td></tr><tr><td>позволяла обработке ввода/вывода прерывать или задерживать данную задачу</td><td>от 8 до 15</td></tr></table></td></tr>
<tr><td>служебные системные операции</td><td>Служебные системные операции – это время, затрачиваемое контроллером на незапланированный обмен данными.
Незапланированный обмен данными – это всякий обмен данными, который вы не конфигурируете посредством папки проекта I/O Configuration, как инструкции Message (MSG) и обмен данными с HMI или рабочими станциями.
Служебные системные операции прерывают только непрерывную задачу.
Квант времени на служебные системные операции определяет долю времени (за исключением времени выполнения периодических и событийных задач), посвящаемую контроллером незапланированному обмену данными.
Контроллер выполняет незапланированный обмен данными в течение максимум 1 мс, после чего возобновляет выполнение непрерывной задачи.</td></tr>
<tr><td>непрерывная задача</td><td>Вам не нужно назначать приоритет непрерывной задаче. Она всегда работает с наименьшим приоритетом. Все остальные задачи прерывают непрерывную задачу.</td></tr>
</table>

ПРИМЕР

В следующем примере показано выполнение проекта с тремя пользовательскими задачами.

Задача:	Приоритет:	Периодичность:	Время выполнения:	Продолжительность:
планировщик перемещений	не применяется	8 мс (частота грубого обновления)	1 мс	1 мс
событийная задача 1	1	не применяется	1 мс	от 1 до 2 мс
периодическая задача 1	2	12 мс	2 мс	от 2 до 4 мс
задача ввода/вывода – не применяется для контроллеров ControlLogix и SoftLogix. См. стр. 4-6.	7	5 мс (наименьший RPI)	1 мс	от 1 до 5 мс
служебные системные операции	не применяется	квант времени = 20%	1 мс	от 1 до 6 мс
непрерывная задача	не применяется	не применяется	20 мс	48 мс



Описание:

①	Сначала контроллер выполняет планировщика перемещений и задачу ввода/вывода (если она имеется).
②	После выполнения непрерывной задачи в течение 4 мс контроллер запускает служебные системные операции.
③	Период для периодической задачи 1 (12 мс) истекает, поэтому эта задача прерывает непрерывную задачу.
④	После выполнения непрерывной задачи в течение еще 4 мс контроллер запускает служебные системные операции.
⑤	Возникают условия запуска (триггеры) для событийной задачи 1. • Событийная задача 1 ожидает выполнения планировщика перемещений. • Задачи с более низкими приоритетами задерживаются больше.
⑥	Непрерывная задача автоматически перезапускается.

Выделение достаточного времени для незапланированного обмена данными

Незапланированный обмен данными происходит лишь в то время, когда не работает ни периодическая, ни событийная задача. Если вы используете несколько задач, убедитесь в том, что их времена сканирования и промежутки между выполнениями обеспечивают достаточное время для незапланированного обмена данными.

Если вы работаете в многозадачном режиме, руководствуйтесь следующими правилами:

1. Время выполнения задачи с самым высоким приоритетом значительно меньше частоты ее обновления.
2. Общее время выполнения всех ваших задач значительно меньше частоты обновления задач с наименьшим приоритетом.

Например, для следующей конфигурации задач:

Задача:	Приоритет:	Время выполнения:	Частота обновления:
1	выше	20 мс	80 мс
2	ниже	30 мс	100 мс
	общее время выполнения:	50 мс	

1. Время выполнения задачи с наивысшим приоритетом (Задача 1) значительно меньше частоты ее обновления (20 мс меньше 80 мс).
2. Общее время выполнения всех задач значительно меньше частоты обновления задачи с наименьшим приоритетом (50 мс меньше 100 мс).

При этом, как правило, остается достаточно времени для незапланированного обмена данными.

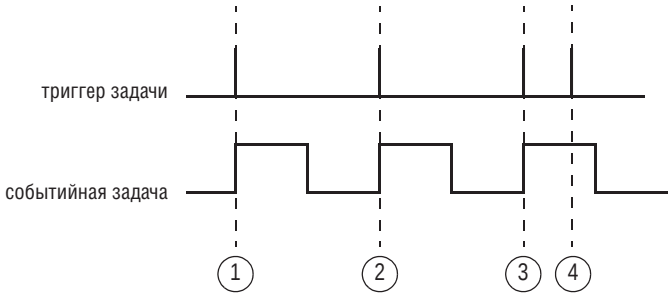
- Откорректируйте частоты обновления задач таким образом, чтобы добиться наилучшего компромисса между выполнением вашей логики и обслуживанием незапланированного обмена данными.
- Если в вашем проекте есть непрерывная задача, то незапланированный обмен данными происходит в течение определенного процента времени контроллера (за исключением времени выполнения периодических и событийных задач). См. «служебные системные операции» на стр. 4-6.

Избежание перекрытий

Перекрытие – это состояние, когда задача (периодическая или событийная) запускается в то время, когда эта задача продолжает выполняться от предыдущего триггера.

ВАЖНО

В случае перекрытия контроллер игнорирует триггер, вызвавший это состояние. Иными словами, может быть пропущено важное выполнение задачи.



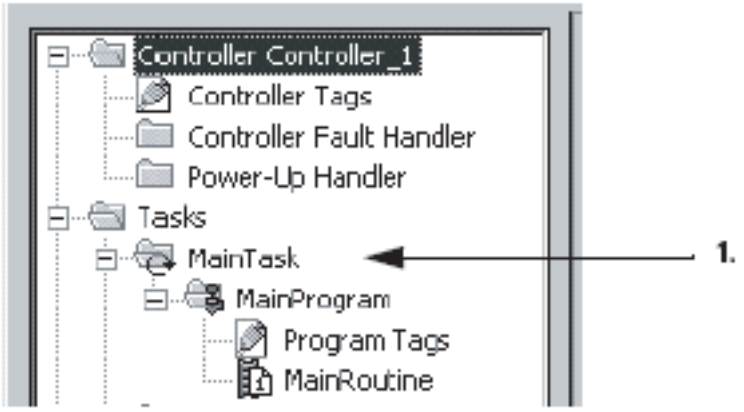
Описание:	
①	Возникает триггер задачи. Задача выполняется.
②	Возникает триггер задачи. Задача выполняется.
③	Возникает триггер задачи. Задача выполняется.
④	Происходит перекрытие. Задача запускается в то время, когда она еще выполняется. Триггер не перезапускает данную задачу. Данный триггер игнорируется.

Каждой задаче требуется достаточное время для ее завершения перед очередным запуском. Убедитесь в том, что время сканирования задачи значительно меньше частоты возникновения триггера. При возникновении перекрытия уменьшите частоту запуска задачи:

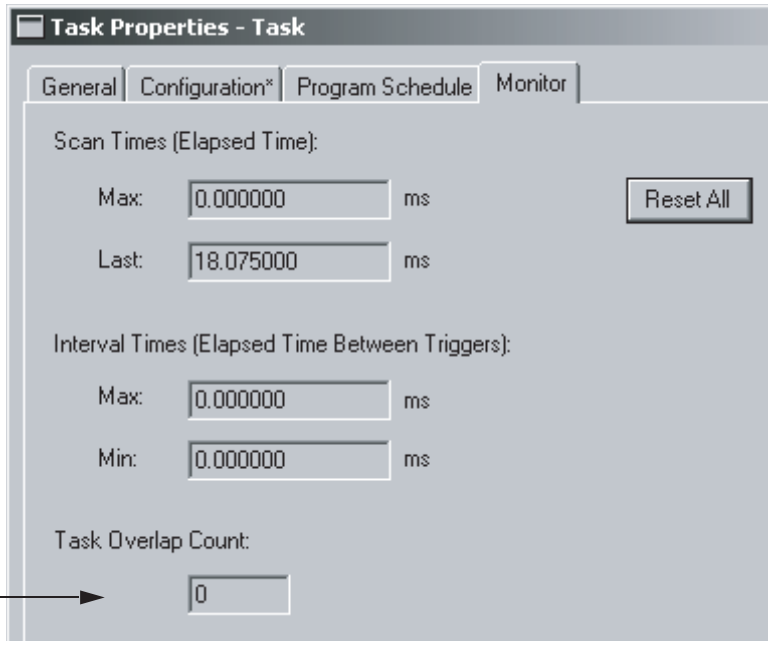
Для этого типа задачи:	Примите следующие меры:
периодическая	увеличьте периодичность задачи
событийная	откорректируйте конфигурацию вашей системы таким образом, чтобы данная задача запускалась с меньшей частотой.

Ручная проверка наличия перекрытий

Чтобы вручную проверить, происходят ли перекрытия для какой-либо задачи:



1. В организаторе контроллера щелкните правой кнопкой мыши по соответствующей задаче и выберите *Properties* (Свойства).
2. Щелкните по закладке *Monitor* (Монитор).



3. Чтобы закрыть это диалоговое окно, нажмите OK.

Программная проверка наличия перекрытий

При возникновении перекрытия контроллер:

- заносит неосновную ошибку в объект FAULTLOG
- сохраняет информацию о перекрытии в объекте TASK для данной задачи

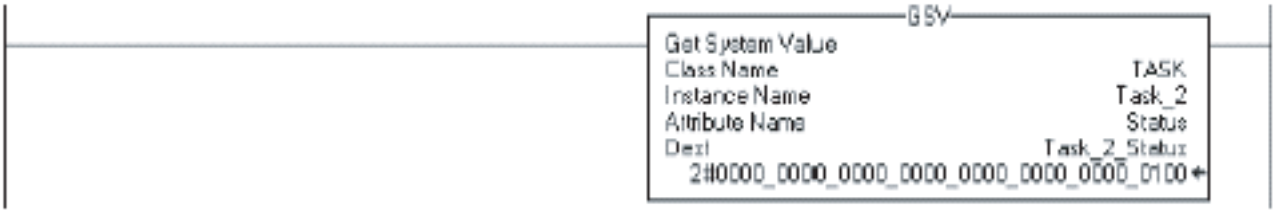
Для написания логики, проверяющей на наличие перекрытия, используйте инструкцию Get System Value (GSV) для контроля одного из следующих объектов:

Если вы хотите:	**То обратитесь к следующему объекту и атрибуту:**			
	Объект:	**Атрибут:**	**Тип данных:**	**Описание:**
определить, произошло ли перекрытие для какой-либо задачи	FAULTLOG	MinorFaultBits	DINT	Отдельные биты, указывающие на неосновную ошибку
				Чтобы определить, имеет ли место следующее: — **Проверьте этот бит:**
				Инструкция произвела неосновную ошибку — 4
				Для какой-либо задачи произошло перекрытие. — *6*
				Последовательный порт произвел неосновную ошибку — 9
				Батерея отсутствует или требует замены — 10
определить, произошло ли перекрытие для определенной задачи	TASK	Status	DINT	Информация о состоянии задачи. Если контроллер установит один из следующих битов, вы должны вручную сбросить соответствующий бит.
				Чтобы определить, имеет ли место следующее: — **Проверьте этот бит:**
				Инструкция EVENT запустила задачу (только для событийных задач). — 0
				Задача запустилась по тайм-ауту (только для событийных задач). — 1
				Для данной задачи произошло перекрытие. — *2*
определить, сколько раз произошло перекрытие	TASK	OverlapCount	DINT	Действует для событийной или периодической задачи. Для сброса счетчика установите этот атрибут на 0.

ПРИМЕР

Программная проверка наличия перекрытий

1. Инструкция GSV устанавливает атрибут *Task_2_Status* =Status (Состояние) для задачи *Task_2* (значение DINT).



2. Если *Task_2_Status.2* = 1, то произошло перекрытие, и следует получить значение счетчика перекрытий:

Инструкция GSV устанавливает атрибут *Task_2_Overlap_Count* (тег DINT) = OverlapCount (Счетчик перекрытий) задачи *Task_2.*

3. Если *Condition_1* = 1, то необходимо сбросить биты атрибута Status для задачи *Task_2.*

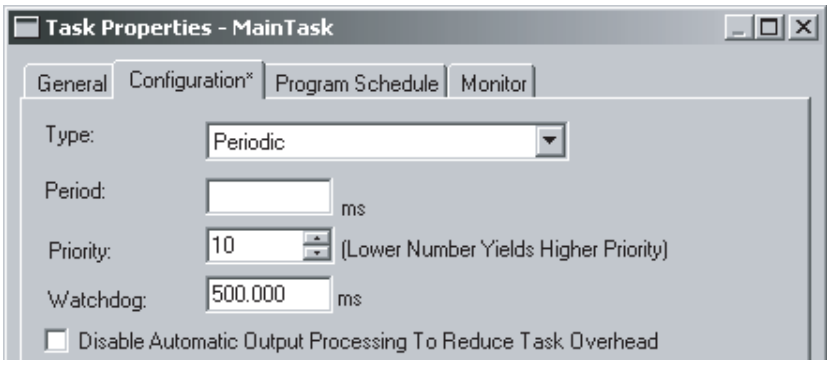
Инструкция SSV устанавливает атрибут Status задачи *Task_2* = *Zero*. Zero – это тег DINT, имеющий значение 0.

Конфигурирование обработки вывода для задачи

В конце выполнения задачи контроллер выполняет служебные операции (обработку вывода) для модулей ввода/вывода в вашей системе. Хотя это *не* то же самое, что обновление этих модулей, такая обработка вывода может оказывать влияние на обновление модулей ввода/вывода в вашей системе.

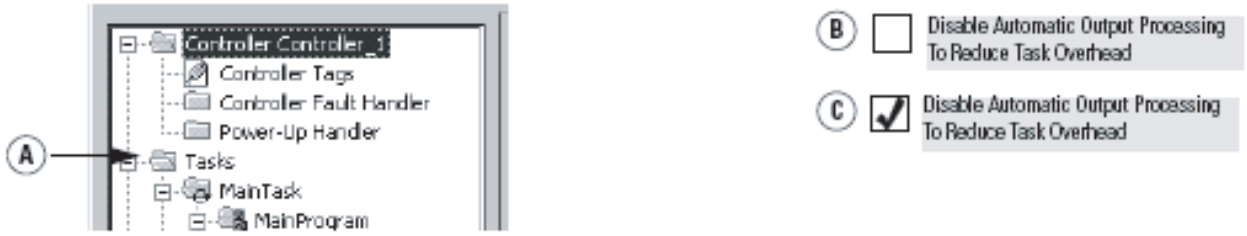
У вас есть возможность отключить эту обработку вывода для определенной задачи, что уменьшает истекшее время выполнения данной задачи.

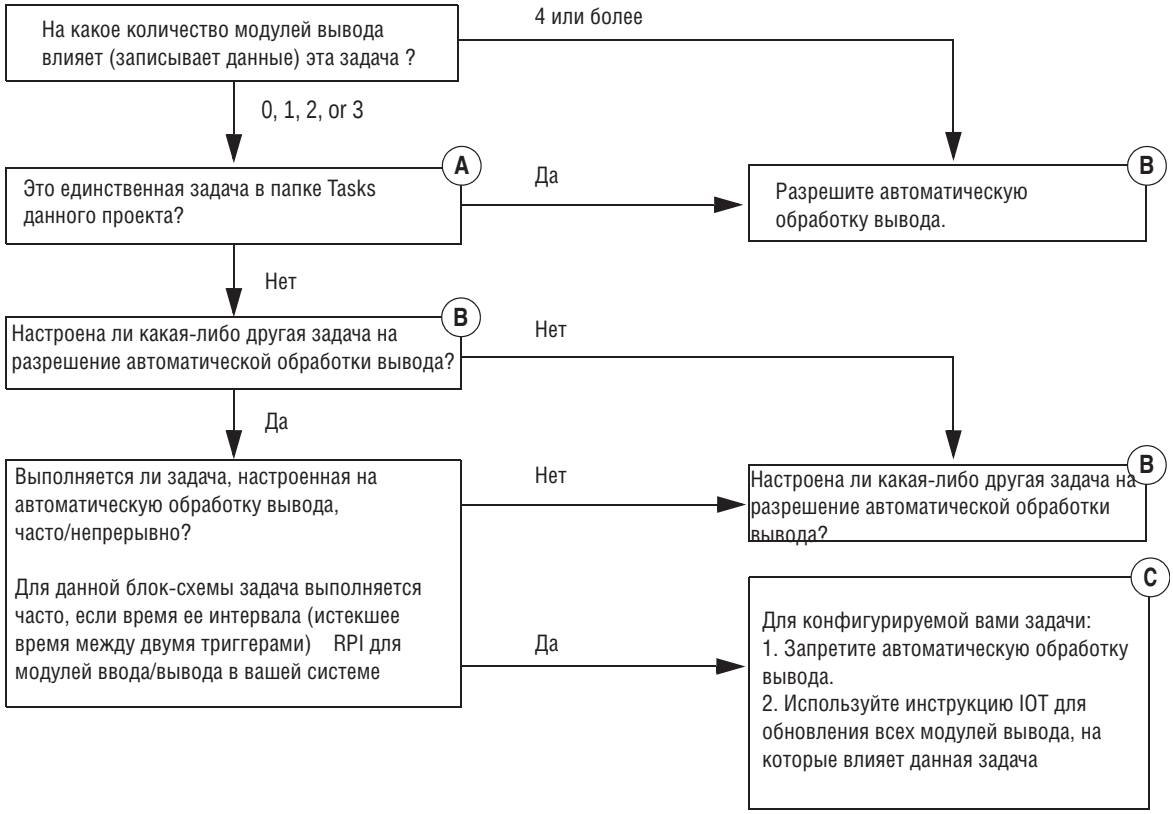
Разрешите или запретите обработку вывода в конце задачи →



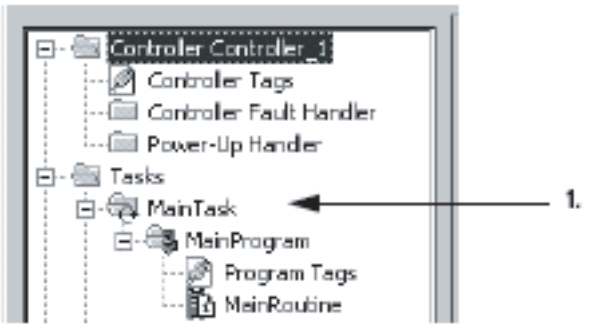
Чтобы выбрать настройку обработки вывода для задачи, воспользуйтесь следующей блок-схемой

Рисунок 4.1 Выбор настройки обработки вывода для задачи.



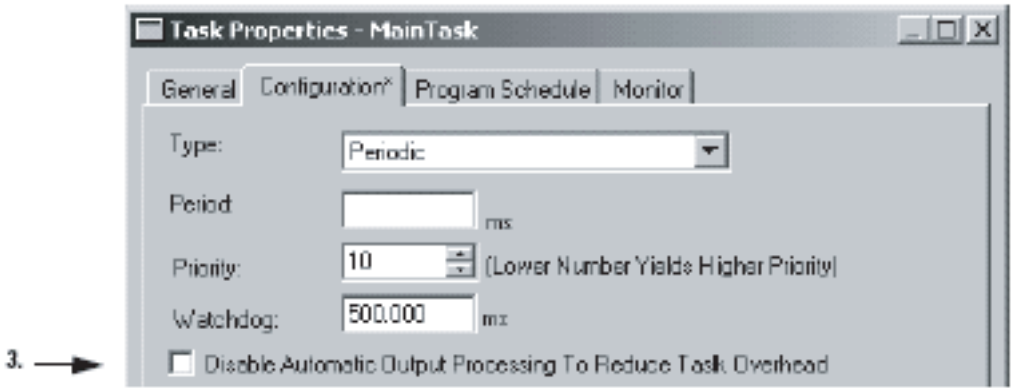


Ручное конфигурирование обработки вывода



1. В организаторе контроллера щелкните правой кнопкой мыши по соответствующей задаче и выберите *Properties* (Свойства).

2. Щелкните по закладке *Configuration* (Конфигурация).



3. Сконфигурируйте обработку вывода для данной задачи:

Если вы хотите:	То:
разрешить обработку вывода в конце выполнения задачи	Снимите флажок *Disable Automatic Output Processing To Reduce Task Overhead* (Запрещение автоматической обработки вывода для уменьшения служебных операций задачи) (настройка по умолчанию).
запретить обработку вывода в конце выполнения задачи	Установите флажок *Disable Automatic Output Processing To Reduce Task Overhead*.

4. Нажмите 

Программное конфигурирование обработки вывода

Чтобы написать логику для конфигурирования обработки вывода для какой-либо задачи, используйте инструкцию Set System Value (SSV). Обращайтесь к следующему атрибуту объекта TASK для данной задачи:

Если вы хотите:	То обратитесь к этому атрибуту:	Тип данных:	Инструкция:	Описание:	
разрешить или запретить	DisableUpdateOutputs	DINT	GSV	Чтобы:	Установите этот атрибут на:
обработку вывода в конце выполнения задачи			SSV	разрешить обработку вывода в конце выполнения задачи	0
				запретить обработку вывода в конце выполнения задачи	1 (или любое ненулевое значение)

ПРИМЕР Программное конфигурирование обработки вывода

Если *Condition_1* = 0, то задаче *Task_2* разрешается обрабатывать вывод по ее завершении.

1. Инструкция ONS ограничивает реальное выполнение инструкции SSV одним сканированием.
2. Инструкция SSV устанавливает атрибут DisableUpdateOutputs задачи *Task_2* = 0. Это позволяет задаче автоматически обрабатывать вывод после завершения ее выполнения.

Если *Condition_1* = 1, то задаче *Task_2* не разрешается обрабатывать вывод по ее завершении.

1. Инструкция ONS ограничивает реальное выполнение инструкции SSV одним сканированием.
2. Инструкция SSV устанавливает атрибут DisableUpdateOutputs задачи *Task_2* = 1. Это запрещает задаче автоматически обрабатывать вывод после завершения ее выполнения.

Запрещение задачи

По умолчанию каждая задача (событийная, периодическая или непрерывная) выполняется от своего триггера. У вас есть возможность воспрепятствовать выполнению задачи при возникновении соответствующего триггера (т.е. запретить задачу). Это полезно при тестировании, диагностике и запуске вашего проекта.

Если вы хотите:	То:
разрешить выполнение задачи при возникновении соответствующего триггера	Снимите запрет с данной задачи (настройка по умолчанию).
запретить выполнение задачи при возникновении соответствующего триггера	Запретите данную задачу.

ПРИМЕР

Запрещение задачи

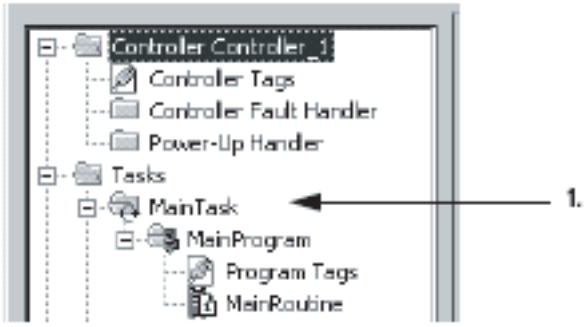
При вводе в эксплуатацию системы, использующей несколько задач, вы можете сначала протестировать каждую задачу в отдельности.

1. Запретите все задачи, кроме одной, и протестируйте эту задачу.
2. Когда задача будет отвечать всем вашим требованиям, запретите ее и снимите запрет с какой-либо другой задачи.
3. Продолжайте этот процесс до тех пор, пока вы не протестируете все свои задачи.

Если задача запрещена, контроллер все равно выполняет предварительное сканирование этой задачи при переходе контроллера из режима программирования в режим выполнения или тестирования.

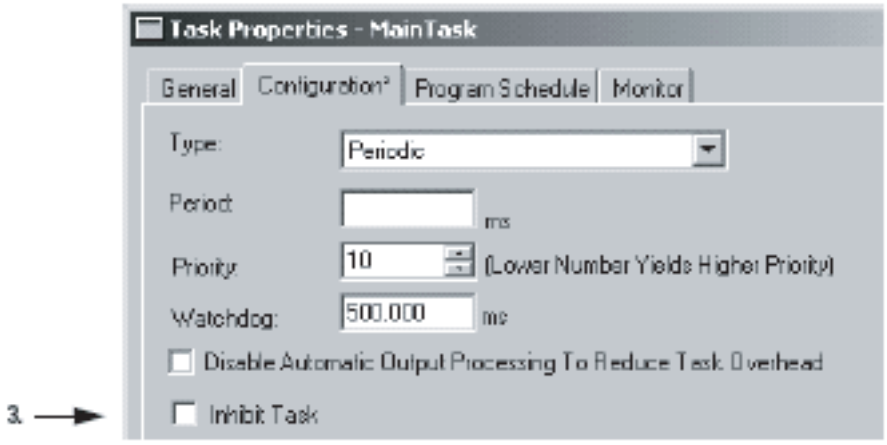
Ручное запрещение или отмена запрещения задачи

Чтобы вручную запретить выполнение задачи или снять запрет с выполнения задачи, используйте диалоговое окно свойств для данной задачи.



1. В организаторе контроллера щелкните правой кнопкой мыши по соответствующей задаче и выберите *Properties* (Свойства).

2. Щелкните по закладке *Configuration* (Конфигурация).



3. Запретите задачу или снимите запрет с задачи:

Если вы хотите:	**То:**
разрешить выполнение задачи при возникновении соответствующего триггера	Снимите флажок *Inhibit Task* (Запрещение задачи) (настройка по умолчанию).
запретить выполнение задачи при возникновении соответствующего триггера	Установите флажок *Inhibit Task*.

4. Нажмите 

Программное запрещение или отмена запрещения задачи

Чтобы написать логику для запрещения или отмены запрещения какой-либо задачи, используйте инструкцию Set System Value (SSV) для обращения к следующему атрибуту объекта TASK для данной задачи:

Атрибут:	Тип данных:	Инструкция:	Описание:	
InhibitTask	DINT	GSV	Не позволяет задаче выполняться.	
		SSV	**Чтобы:**	**Установите этот атрибут на:**
			разрешить выполнение задачи	0 (по умолчанию)
			запретить выполнение задачи	1 (или любое ненулевое значение)

ПРИМЕР Программное запрещение или отмена запрещения задачи

Если *Condition_1* = 0, то выполнение задачи *Task_2* разрешается.

1. Инструкция ONS ограничивает реальное выполнение инструкции SSV одним сканированием.
2. Инструкция SSV устанавливает атрибут InhibitTask задачи *Task_2* = 0. Это отменяет запрет на выполнение задачи.

Если *Condition_1* = 1, то выполнение задачи *Task_2* запрещается.

1. Инструкция ONS ограничивает реальное выполнение инструкции SSV одним сканированием.
2. Инструкция SSV устанавливает атрибут InhibitTask задачи *Task_2* = 1. Это запрещает выполнение задачи.

Выбор триггера для событийной задачи

Если событийная задача сконфигурирована правильно, она прерывает все остальные задачи на минимальное время, требуемое для отклика на событие. Каждой событийной задаче требуется конкретный триггер, определяющий, когда задача должна выполняться.

Для запуска событийной задачи при:	Используйте этот триггер:	С учетом следующих соображений:
включении или выключении цифрового ввода	Module Input Data State Change	• Только один модуль ввода может запускать конкретную событийную задачу. • Модуль ввода запускает событийную задачу на основе конфигурации изменения состояния (change of state – COS) для данного модуля. Конфигурация COS определяет, какие точки при включении или выключении побуждают модуль к производству данных. Такое производство данных, вызванное COS, запускает событийную задачу. • Как правило, COS следует разрешать только для одной точки модуля. Если вы разрешите COS для нескольких точек, может произойти перекрытие событийной задачи.
выборке данных аналоговым модулем	Module Input Data State Change	• Только один модуль ввода может запускать конкретную событийную задачу. • Аналоговый модуль запускает событийную задачу после каждой выборки реального времени (real time sample –RTS) по каналам. • Все каналы модуля используют одну и ту же RTS.
получении контроллером новых данных посредством потребляемого тега	Consumed Tag	• Только один потребляемый тег может запускать конкретную событийную задачу. • В общем случае используйте инструкцию IOT в производящем контроллере для подачи сигнала о производстве новых данных. Инструкция IOT устанавливает срабатывающий по событию триггер в производящем теге. Этот триггер переходит к потребляемому тегу и запускает событийную задачу. • Когда потребляемый тег запускает событийную задачу, эта событийная задача начинает выполняться только после поступления всех данных.
включении или выключении регистрационного входа для оси	Axis Registration 1 или 2	• Чтобы регистрационный вход запускал событийную задачу, сначала выполните инструкцию Motion Arm Registration (MAR). Это позволяет соответствующей оси определять регистрационный вход и, в свою очередь, запускать событийную задачу. • После запуска регистрационным входом событийной задачи вновь выполните инструкцию MAR, чтобы активизировать ось для следующего регистрационного входа. • Если ваша логика сканируется *не* достаточно быстро для активизации оси для очередного регистрационного входа, вы можете включить инструкцию MAR непосредственно в событийную задачу.
достижении осью положения, заданного в качестве контрольной точки	Axis Watch	• Чтобы контрольное положение запускало событийную задачу, сначала выполните инструкцию Motion Arm Watch (MAW). Это позволяет соответствующей оси определять контрольное положение и, в свою очередь, запускать событийную задачу. • После запуска контрольным положением событийной задачи вновь выполните инструкцию MAW, чтобы активизировать ось для следующего контрольного положения. • Если ваша логика сканируется *не* достаточно быстро для активизации оси для очередного контрольного положения, вы можете включить инструкцию MAW непосредственно в событийную задачу
завершении выполнения планировщика перемещений	Motion Group Execution	• Период грубого обновления для группы перемещения запускает выполнение как планировщика перемещений, так и событийную задачу. • Поскольку планировщик перемещений прерывает все остальные задачи, он выполняется первым. Если вы назначите событийной задаче наивысший приоритет, она будет выполняться после планировщика перемещений.
наступлении определенного условия или условий в логике программы	инструкция EVENT	Одна и та же задача может запускаться несколькими инструкциями EVENT. Это позволяет выполнять задачи из других программ.

Далее в качестве примера приводится ряд случаев использования событийных задач и соответствующих триггеров:

В этом случае:	**Используйте событийную задачу с этим триггером:**
На упаковочной линии производится склеивание коробок. При поступлении коробки на операцию склеивания контроллер должен немедленно выполнить процедуру склеивания	Module Input Data State Change
На производственной линии используется датчик присутствия для определения наличия детали. Поскольку датчик присутствия находится во включенном состоянии лишь очень короткое время (импульс), непрерывная задача может пропустить переход датчика из выключенного состояния во включенное.	Module Input Data State Change
На испытательном стенде двигателя вы должны получать и архивировать каждую выборку аналоговых данных.	Module Input Data State Change
Контроллер А производит массив производственных данных для Контроллера Б. Вы хотите сделать так, чтобы Контроллер Б не использовал значения в то время, когда Контроллер А обновляет массив.	Consumed Tag
На линии упаковки конфет необходимо обеспечить перфорацию каждой конфеты в заданном месте. При каждом обнаружении датчиком регистрации соответствующей отметки следует проверить точность оси и при необходимости выполнить ее корректировку.	Axis Registration 1 или 2
На участке наклейки этикеток на бутылки вы хотите проверять положение этикетки на бутылке. Проверка этикетки осуществляется по достижении осью положения, заданного в качестве контрольной точки.	Axis Watch
Станция склеивания должна корректировать количество наносимого клея для компенсации изменений скорости перемещения оси. После выполнения планировщика перемещения необходимо проверить заданную командой скорость перемещения оси и при необходимости изменить количество клея.	Motion Group Execution
Вся производственная линия должна быть остановлена при обнаружении какой-либо программой опасного состояния. Процедура останова не зависит от типа опасного состояния.	инструкция EVENT

Триггеры, которые вы можете использовать для событийной задачи, зависят от типа вашего контроллера Logix5000.

ВАЖНО

Программное обеспечение RSLogix 5000 может позволить вам сконфигурировать не поддерживаемый вашим контроллером триггер для событийной задачи. При этом проект будет проверен и успешно загрузится, но соответствующая событийная задача выполняться не будет.

Таблица 4.1 Используйте следующую таблицу, чтобы определить, какие типы триггеров событийной задачи поддерживаются различными контроллерами Logix5000

Если у вас этот контроллер:	**Вы можете использовать эти триггеры событийной задачи:**					
	Module Input Data State Change	Consumed Tag	Axis Registration 1 или 2	Axis Watch	Motion Group Execution	Инструкция EVENT
CompactLogix		✓				✓
FlexLogix		✓				✓
ControlLogix	✓	✓	✓	✓	✓	✓
DriveLogix		✓	✓	✓	✓	✓
SoftLogix5800	✓(1)	✓(2)	✓	✓	✓	✓

(1) Требуется модуль ввода/вывода 1756 или виртуальная объединительная плата
(2) Контроллер SoftLogix5800 производит и потребляет теги только в сети ControlNet.

Использование изменения состояния данных модуля ввода в качестве триггера

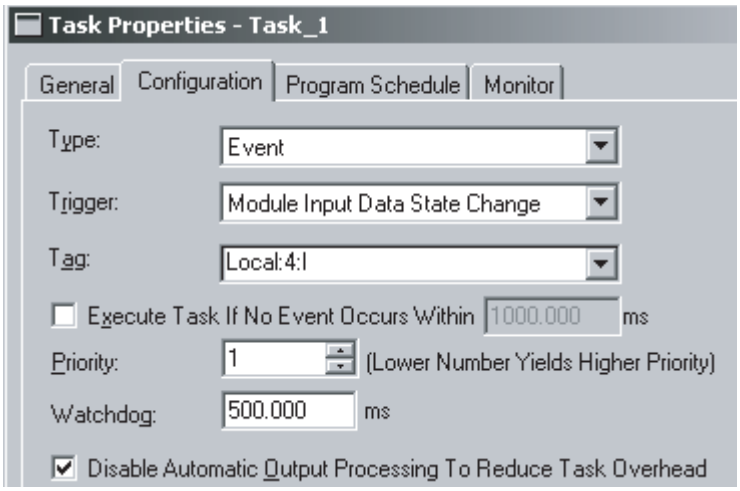
Для запуска событийной задачи на основе данных, поступающих от модуля ввода, используйте триггер *Module Input Data Change* (Изменение данных модуля ввода).

Данная задача запускается событием.

Данная задача запускается на основе данных модуля ввода.

Данная задача запускается этим триггером в виде тега ввода.

По завершении выполнения задачи цифровой вывод в локальном шасси не обновляется.



Как модуль ввода/вывода запускает событийную задачу

Следующие термины применяются при описании работы модуля ввода:

Термин:	Определение:
многоадресная передача	Механизм, при котором модуль отправляет по сети данные, получаемые одновременно несколькими приемниками (устройствами). Характеризует особенность линии ввода/вывода Logix5000, поддерживающей одновременное получение входных данных несколькими контроллерами от одного модуля ввода/вывода.
запрашиваемый межпакетный интервал (RPI)	RPI определяет интервал, с которым модуль осуществляет многоадресную передачу своих данных. Например, модуль ввода направляет данные в контроллер с RPI, заданным вами для этого модуля. • Допустимый диапазон – от 0,2 мс (200 микросекунд) до 750 мс. • По истечении заданного времени модуль осуществляет многоадресную передачу данных. Также это называется циклическим обновлением.
выборка реального времени (real time sample – RTS)	RTS определяет, когда аналоговый модуль должен сканировать свои каналы и осуществлять многоадресную передачу данных (обновлять буфер ввода, а затем осуществлять групповую рассылку). • *RPI* определяет, когда модуль должен осуществлять многоадресную передачу текущего содержимого буфера ввода без сканирования (обновления) каналов. • Модуль каждый раз сбрасывает таймер RPI, и происходит передача RTS.

Термин:	Определение:
изменение состояния (change of state – COS)	Параметр COS указывает цифровому модулю ввода осуществлять многоадресную передачу данных всякий раз, когда определенная точка входа переходит из включенного состояния в выключенное (On → Off) или из выключенного состояния во включенное (Off → On). • COS устанавливается для каждой точки в отдельности. • Когда любая точка, для которой установлен COS, получает соответствующее изменение, модуль осуществляет многоадресную передачу данных для всех своих точек. • По умолчанию COS активирован для всех точек как для изменений On → Off, так и для изменений Off → On. • Вы должны указать RPI независимо от того, активируете ли вы COS. Если в пределах интервала RPI не происходит никакого изменения, то модуль передает данные по истечении RPI.

В следующей таблице указывается, в каких случаях модуль ввода осуществляет многоадресную передачу своих данных и запускает событийную задачу *в пределах своего шасси.*

Если модуль ввода является:	И:	То он осуществляет многоадресную рассылку:	И запускает событийную задачу:
цифровым	COS активирован для какой-либо точки модуля	• при получении заданного изменения любой точкой, для которой активирован COS • с интервалом RPI	при получении заданного изменения любой точкой, для которой активирован COS
	COS не активирован ни для какой точки модуля	с интервалом RPI	никогда
аналоговым	RTS ≤ RPI	с интервалом RTS (последнее обновление данных канала)	с интервалом RTS для данного модуля
	RTS > RPI	• с интервалом RTS (последнее обновление данных канала) • с интервалом RPI (не содержит обновленные данные каналов)	с интервалом RTS для данного модуля

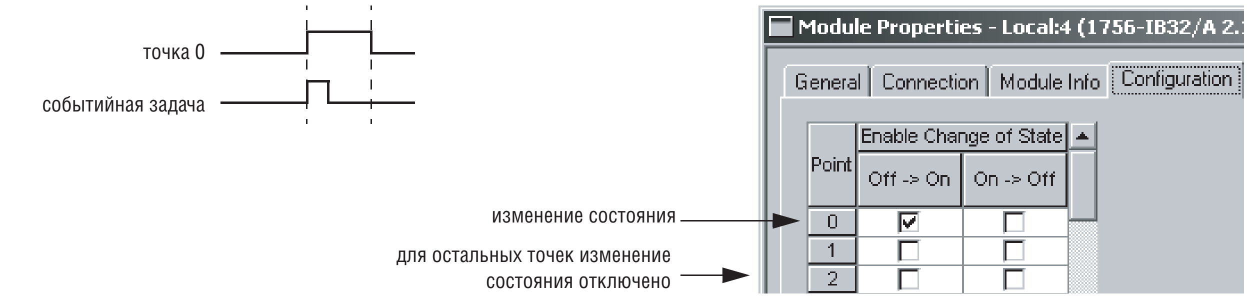
Если модуль находится в удаленном шасси, то только RPI определяет, когда данный контроллер будет получать по сети данные и триггер для событийной задачи.

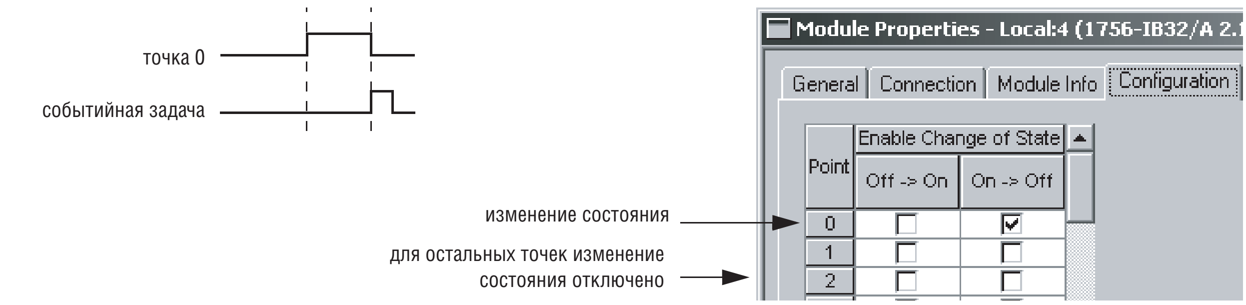
По этой сети:	Контроллер получает данные:
EtherNet/IP	с интервалом, в среднем близким к RPI
ControlNet	с фактическим межпакетным интервалом (≤ RPI)

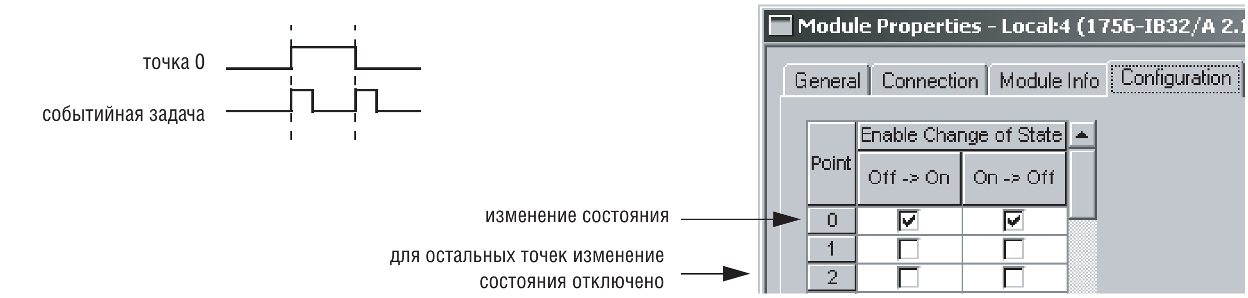
Далее приводится несколько примеров использования COS и RTS:

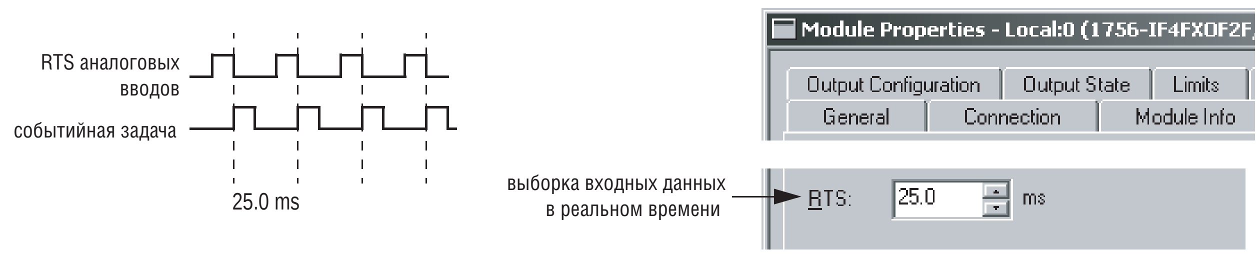
ВАЖНО Если вы используете цифровой модуль для запуска событийной задачи, сконфигурируйте только одну точку модуля для COS. Если вы сконфигурируете несколько точек, то может произойти перекрытие задач.

Ели вы хотите иметь следующее:	То сконфигурируйте модуль ввода следующим образом (точка 0 используется лишь для примера):









Обеспечьте возможность запуска событийной задачи вашим модулем

Чтобы можно было использовать модуль ввода для запуска событийной задачи, этот модуль должен поддерживать запуск событийных задач. Если данный модуль находится в удаленном пункте, соответствующие коммуникационные модули также должны поддерживать запуск событийных задач.

В следующей таблице перечисляются модули Rockwell Automation, протестированные нами на предмет запуска событийных задач. Некоторые модули других поставщиков также могут поддерживать запуск событийных задач. Прежде чем использовать модуль стороннего поставщика, убедитесь в его надлежащей работе.

Категория	Модуль	Категория	Модуль
1756 Discrete	1756-IA16	1756 Analog	1756-IF16
	1756-IA16I		1756-IF4FXOF2F/A
	1756-IA8D		1756-IF6CIS
	1756-IB16		1756-IF6I
	1756-IB16D		1756-IF8
	1756-IB16I		1756-IR6I
	1756-IB32/A		1756-IT6I
	1756-IB32/B		1756-IT6I2
	1756-IC16	1756 Generic	1756-MODULE
	1756-IH16I	1756 Communication	1756-CNB/A
	1756-IM16I		1756-CNB/B
	1756-IN16		1756-CNB/D
	1756-IV16/A		1756-CNBR/A
	1756-IV32/A		1756-CNBR/B
			1756-CNBR/D
			1756-DNB
			1756-ENBT/A
			1756-SYNCH/A
		SoftDNB	1784-PCIDS/A
		1789 Generic	1789-MODULE

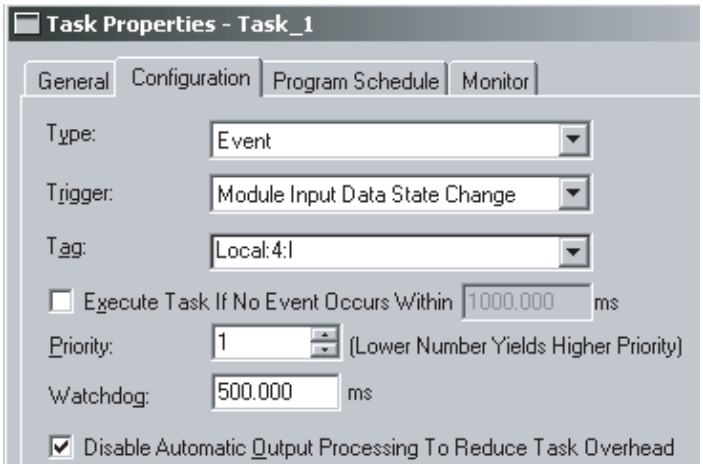
Контрольный перечень для задачи, запускаемой от события ввода

Для этого параметра:	Обеспечьте выполнение следующих требований:
❒ 1. Тип модуля ввода	Для максимального быстродействия используйте следующие модули: • Для максимального цифрового быстродействия используйте модуль 1756-IB32/B. • Для максимального аналогового быстродействия используйте модуль 1756-IF4FXOF2F
❒ 2. Местоположение модуля ввода/вывода	Поместите модуль, запускающий событие, и модули, откликающиеся на это событие (выводы), в то же шасси, где находится контроллер. Для удаленных модулей ко времени отклика добавляется сетевой обмен.
❒ 3. Количество локальных модулей	Ограничьте количество модулей в локальном шасси. Дополнительные модули увеличивают вероятность задержек в объединительной плате.
❒ 4. Изменение состояния (COS)	Если событие запускается цифровым устройством, то активируйте COS лишь для той точки, которая запускает соответствующую событийную задачу. • Активируйте изменение состояния для типа перехода, запускающего задачу: Off → On, On → Off или и того, и другого. • Если вы сконфигурируете COS и для Off → On, и для On → Off, то данная точка будет запускать событийную задачу при всяком ее включении и выключении. Убедитесь в том, продолжительность ввода больше времени сканирования задачи. В противном случае может произойти перекрытие. • Отключите COS (снимите флажок) для остальных точек модуля ввода. Если вы сконфигурируете для COS несколько точек модуля, то каждая из них сможет запускать событийную задачу. Это может привести к перекрытию.
❒ 5. Приоритет задачи	Сконфигурируйте событийную задачу как самую высокоприоритетную. Если какая-либо периодическая задача имеет более высокий приоритет, событийная задача может быть вынуждена ожидать завершения выполнения такой периодической задачи.
❒ 6. Планировщик перемещений	Планировщик перемещений прерывает все остальные задачи независимо от их приоритетов. • Количество осей и период грубого обновления для группы перемещения влияют на частоту и длительность выполнения планировщика перемещений. • Если планировщик перемещений выполняется во время запуска задачи, эта задача будет ожидать окончания выполнения планировщика перемещений. • Если в процессе выполнения какой-либо задачи наступает срок грубого обновления, то задача приостанавливается на время выполнения планировщика перемещений.
❒ 7. Количество событийных задач	Ограничивайте количество событийных задач. Каждая дополнительная задача уменьшает время обработки, отводимое другим задачам. Это может привести в перекрытию.
❒ 8. Автоматическая обработка вывода	Для событийной задачи вы, как правило, можете отключить автоматическую обработку вывода (настройка по умолчанию). Это уменьшает истекшее время выполнения задачи. Для принятия решения по этой настройке воспользуйтесь рисунком 4.1 на странице 4-14.
❒ 9. Инструкция IOT	Используйте инструкцию IOT для каждого модуля вывода, к которому вы обращаетесь в событийной задаче. Инструкция IOT отменяет RPI для данного модуля и осуществляет немедленную передачу данных.

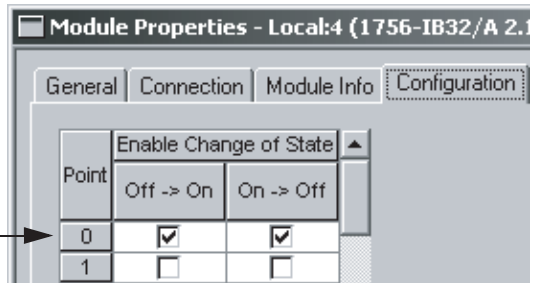
ПРИМЕР

При прохождении деталей мимо сбрасывателя контроллер должен принять решение о том, следует ли включать сбрасыватель. Если сбрасыватель включен, контроллер также должен выключить его до поступления на это место следующей детали. Из-за скорости линии сбрасыватель управляется событийной задачей.

Фотоглаз, находящийся в месте установки сбрасывателя, фиксирует нахождение детали в этом месте. Вход подключен к модулю в слоте 4 локального шасси.

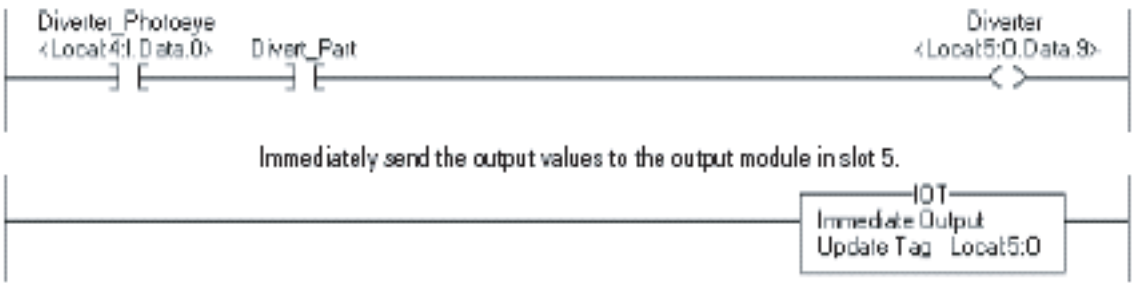


Фотоглаз сбрасывателя (точка 0) настроен на изменение состояния "выключено" (Off) и "включено" (On). Это позволяет фотоглазу запускать событийную задачу и при его включении, и при его выключении.



Эта событийная задача использует следующую логику для управления сбрасывателем.

If Diverter_Photoeye = 1 (деталь находится в месте установки сбрасывателя)
And Divert_Part = 1 (сбросить эту деталь)
Then Diverter = 1 (включить сбрасыватель)
Otherwise Diverter = 0 (выключить сбрасыватель)



Оценка времени цикла

Для оценки времени цикла от входа до выхода (от винтика до винтика) воспользуйтесь следующей таблицей:

Фактор:		**Значение:**
1. Каково время входного фильтра для модуля, запускающего данную событийную задачу? Обычно оно указывается в миллисекундах. Переведите его в микросекунды (µc).		µc
2. Каково время аппаратного отклика для модуля ввода, запускающего данную событийную задачу? Убедитесь в том, что вы используете надлежащий тип перехода (Off → On или On → Off). См. таблицу 4.2 на стр. 4-29.		µc
3. Каково время коммуникации через объединительную плату? **Если размер шасси:**	**Используйте это значение (худший случай):**	
4 слота	13 µc	
7 слотов	22 µc	
10 слотов	32 µc	
13 слотов	42 µc	
17 слотов	54 µc	µc
4. Каково общее время выполнения программ событийной задачи?		µc
5. Каково время коммуникации через объединительную плату? (То же значение, что и в п.3.)		µc
6. Каково время аппаратного отклика для модуля вывода?		µc
7. Сложите значения по пунктам с 1 по 6. Это минимальная оценка времени цикла *без учета* задержек и прерываний событийной задачи из-за выполнения планировщика перемещений или других задач.		µc
8. Каково время сканирования группы перемещения?		µc
9. Каково общее время сканирования задач, приоритет которых выше приоритета данной событийной задачи (если таковые имеются)?		µc
10. Сложите значения по пунктам с 7 по 9. Это оценка номинального времени цикла с учетом задержек и прерываний событийной задачи из-за выполнения планировщика перемещений или других задач.		

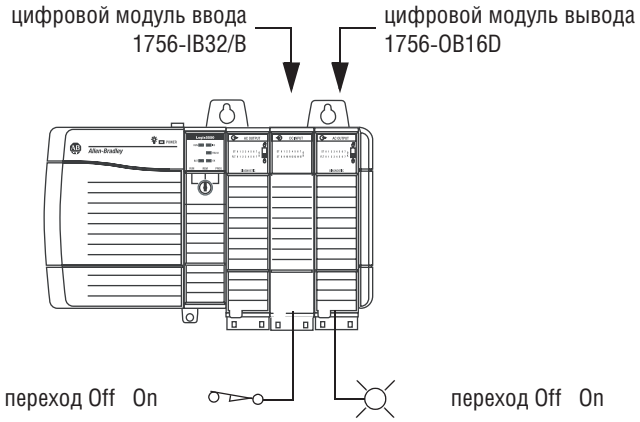
Таблица 4.2 Используйте следующую таблицу для определения номинального времени аппаратного отклика для отдельных модулей ввода/вывода 1756.

Модуль	Номинальное время отклика в мкс:							
	25°C				60°C			
	Off	On	On	Off	Off	On	On	Off
1756-IB16	265		582		265		638	
1756-IB16D	303		613		305		673	
1756-IB32/B	330		359		345		378	
1756-IV16	257		435		254		489	
1756-IV32	381		476		319		536	
1756-OB16D	48		519		51		573	
1756-OB16E	60		290		61		324	
1756-OB32	38		160		49		179	
1756-OV16E	67		260		65		326	
1756-OV32E	65		174		66		210	

ПРИМЕР

Оценка времени цикла

В следующем примере время цикла оценивается для показанной ниже системы. В этом примере временем цикла является время от включения ввода до включения вывода.



Фактор:		**Значение:**
1. Каково время входного фильтра для модуля, запускающего данную событийную задачу? Обычно оно указывается в миллисекундах. Переведите его в микросекунды (µc).		0 µc
2. Каково время аппаратного отклика для модуля ввода, запускающего данную событийную задачу? Убедитесь в том, что вы используете надлежащий тип перехода (Off → On или On → Off). См. таблицу 4.2 на стр. 4-29.		330 µc
3. Каково время коммуникации через объединительную плату?		
Если размер шасси:	**Используйте это значение (худший случай):**	
4 слота	13 µc	
7 слотов	22 µc	
10 слотов	32 µc	
13 слотов	42 µc	
17 слотов	54 µc	13 µc
4. Каково общее время выполнения программ событийной задачи?		400 µc
5. Каково время коммуникации через объединительную плату? (То же значение, что и в п.3.)		13 µc
6. Каково время аппаратного отклика для модуля вывода?		51 µc
7. Сложите значения по пунктам с 1 по 6. Это минимальная оценка времени цикла *без учета* задержек и прерываний событийной задачи из-за выполнения планировщика перемещений или других задач.		807 µc
8. Каково время сканирования группы перемещения?		1130 µc
9. Каково общее время сканирования задач, приоритет которых выше приоритета данной событийной задачи (если таковые имеются)?		0 µc
10. Сложите значения по пунктам с 7 по 9. Это оценка номинального времени цикла с учетом задержек и прерываний событийной задачи из-за выполнения планировщика перемещений или других задач.		1937 µc

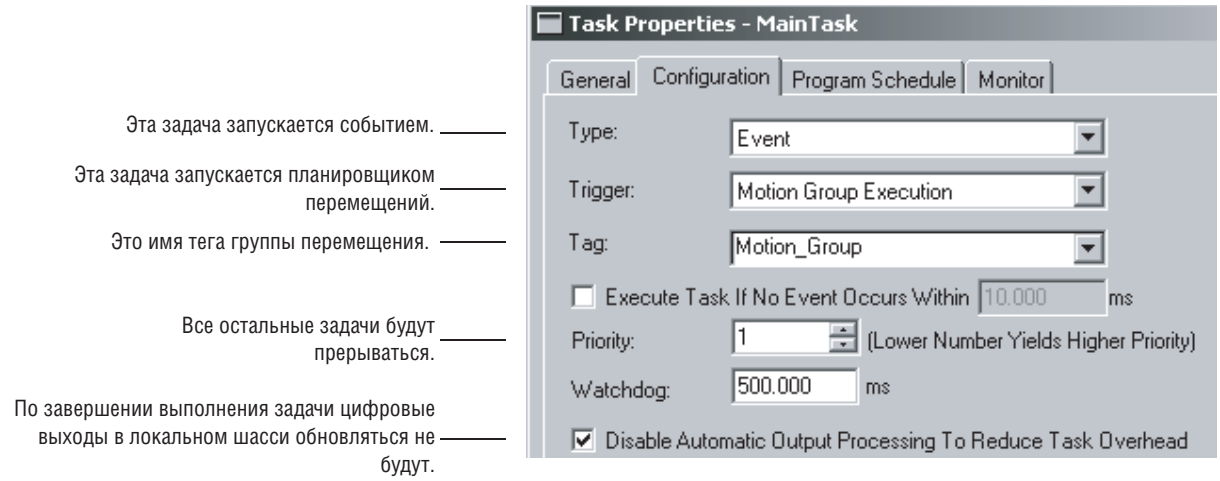
Дополнительные факторы

Следующие факторы влияют на время сканирования для событийной задачи, что в свою очередь влияет на скорость ее отклика на входной сигнал.

Фактор:	Описание:
объем кода в событийной задаче	Каждый элемент логики (цепь, инструкция, конструкция структурированного текста и т.д.) увеличивает время сканирования задачи.
приоритет задачи	Если данная событийная задача не является самой высокоприоритетной, то задача с более высоким приоритетом может задержать или прервать выполнение данной задачи.
инструкции CPS и UID	Если активна одна из этих инструкций, то событийная задача не может прерывать выполняющуюся в данный момент задачу. (Задачу с CPS или UID.)
коммуникационные прерывания	Следующие действия контроллера прерывают задачу независимо от ее приоритета: • обмен данными с модулями ввода/вывода Модули с большими пакетами данных, такие как 1756-DNB, оказывают большее влияние. • обмен данными с последовательным портом

Использование группы перемещения в качестве триггера

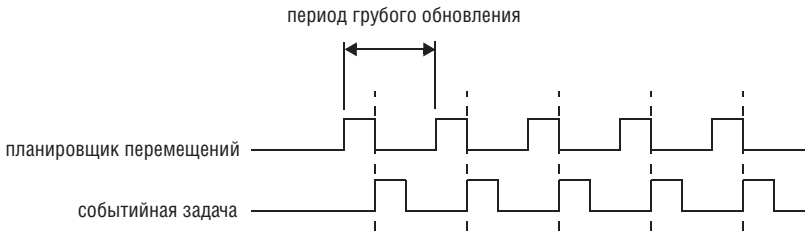
Чтобы совместить выполнение событийной задачи с выполнением планировщика перемещений, используйте триггер *Motion Group Execution* (Выполнение группы перемещения).

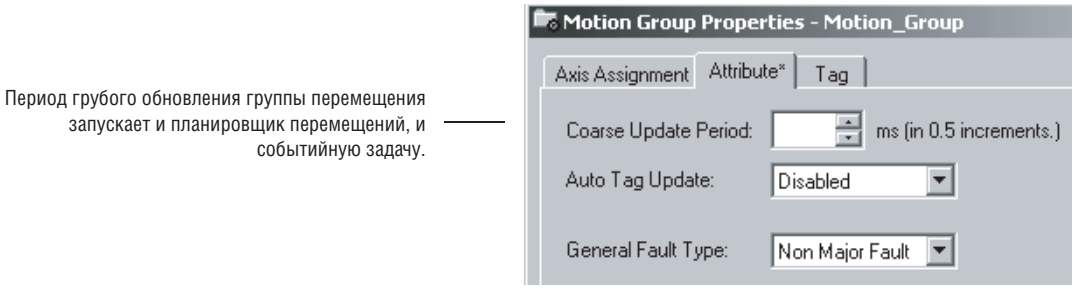


Триггер *Motion Group Execution* работает следующим образом:

- Период грубого обновления группы перемещения запускает выполнение и планировщика перемещений, и событийной задачи.
- Поскольку планировщик перемещений прерывает все остальные задачи, он выполняется первым. Если вы назначите событийной задаче наивысший приоритет, то она будет выполняться сразу же за планировщиком перемещений.

На следующей временной диаграмме показано соотношение между планировщиком перемещений и событийной задачей.



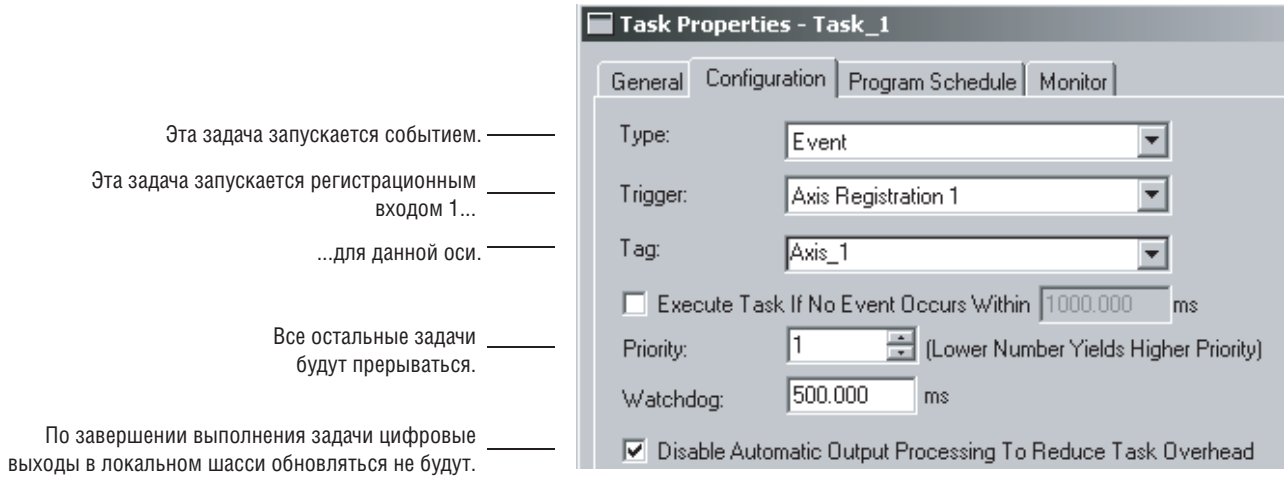


Контрольный перечень для задачи, запускаемой от группы перемещения

Для этого параметра:	Обеспечьте выполнение следующих требований:
❑ 1. Время сканирования	Убедитесь в том, что время сканирования для событийной задачи значительно меньше периода грубого обновления группы перемещения. В противном случае может произойти перекрытие задач.
❑ 2. Приоритетность задачи	Назначьте данной событийной задаче наивысший приоритет. Если какая-либо периодическая задача имеет более высокий приоритет, событийная задача может быть вынуждена ожидать завершения выполнения такой периодической задачи.
❑ 3. Количество событийных задач	Ограничивайте количество событийных задач. Каждая дополнительная задача уменьшает время обработки, отводимое для других задач. Это может привести к перекрытию.
❑ 4. Автоматическая обработка вывода	Для событийной задачи вы, как правило, можете отключить автоматическую обработку вывода (настройка по умолчанию). Это уменьшает истекшее время выполнения задачи. Для принятия решения по этой настройке воспользуйтесь рисунком 4.1 на странице 4-14.

Использование регистрации оси в качестве триггера

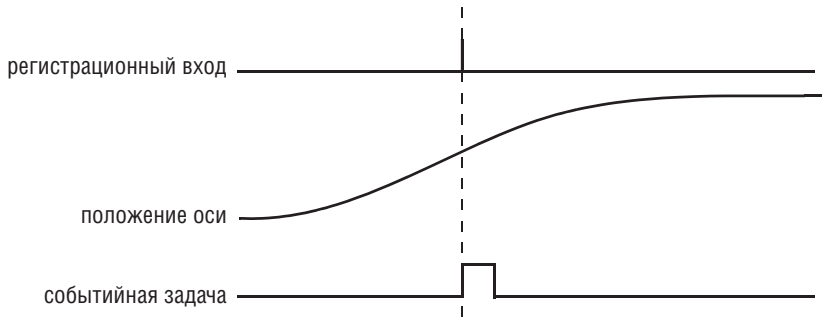
Чтобы событийная задача запускалась регистрационным входом для оси, используйте триггер *Axis Registration (1 или 2)* (Регистрация оси).



По достижении указанным регистрационным входом условий его срабатывания он запускает соответствующую событийную задачу.

- В настройках событийной задачи укажите, какой регистрационный вход должен запускать эту задачу. Выберите *Axis Registration 1* или *Axis Registration 2*.
- Сначала вы должны активизировать регистрационный вход при помощи инструкции Motion Arm Registration (MAR).
- В инструкции MAR операнд Trigger Condition (Условия срабатывания) определяет, какой переход регистрационного входа (Off → On или On → Off) будет запускать данную событийную задачу.
- После запуска задачи регистрационным входом вы должны вновь активизировать данный регистрационный вход.

Следующая временная диаграмма показывает взаимосвязь между регистрационным входом и событийной задачей.



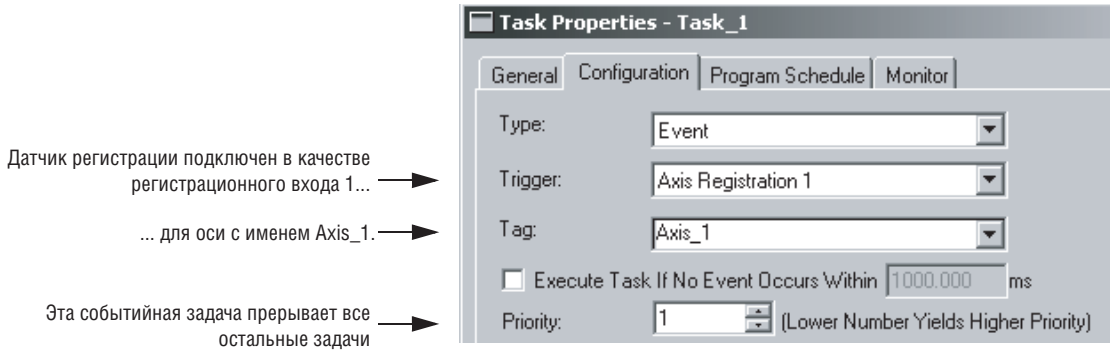
Контрольный перечень для задачи, запускаемой от входа регистрации оси

Для этого параметра:	Обеспечьте выполнение следующих требований:	
❑ 1. Регистрационный вход	Активизируйте регистрационный вход (инструкция MAR). Это позволяет оси обнаруживать регистрационный вход и запускать событийную задачу. • Сначала активизируйте регистрационный вход для обнаружения первого состояния срабатывания. • Повторно активизируйте регистрационный вход после каждого выполнения событийной задачи. • Активизируйте регистрационный вход достаточно быстро для обнаружения каждого состояния срабатывания.	
	Если ваша обычная логика:	**То:**
	работает достаточно быстро для повторной активизации регистрационного входа между состояниями срабатывания Например, ваша обычная логика всегда выполняет как минимум 2 сканирования между регистрационными входами.	При желании вы можете активизировать регистрационный вход в вашей обычной логике.
	работает *недостаточно* быстро для повторной активизации регистрационного входа	Активизируйте регистрационного вход в соответствующей событийной задаче.
❑ 2. Приоритетность задачи	Назначьте данной событийной задаче наивысший приоритет. Если какая-либо периодическая задача имеет более высокий приоритет, событийная задача может быть вынуждена ожидать завершения выполнения такой периодической задачи.	
❑ 3. Количество событийных задач	Ограничивайте количество событийных задач. Каждая дополнительная задача уменьшает время обработки, отводимое для других задач. Это может привести к перекрытию.	
❑ 4. Автоматическая обработка вывода	Для событийной задачи вы, как правило, можете отключить автоматическую обработку вывода (настройка по умолчанию). Это уменьшает истекшее время выполнения задачи. Для принятия решения по этой настройке воспользуйтесь рисунком 4.1 на странице 4-14.	

ПРИМЕР

На линии упаковки конфет необходимо обеспечить перфорацию каждой конфеты в заданном месте.

- При каждом обнаружении датчиком регистрации соответствующей отметки следует проверить точность оси и при необходимости выполнить ее корректировку.
- Из-за скорости линии регистрационный вход должен активизироваться в событийной задаче.



Следующая логика активизирует и реактивизирует регистрационный вход.

Непрерывная задача

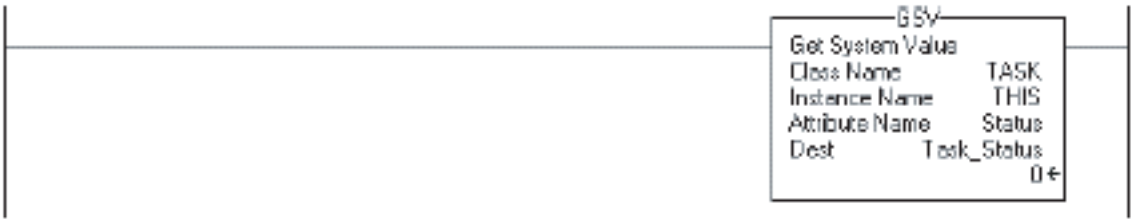
Если *Arm_Registration* = 1 (система готова к поиску регистрационной отметки), то

Инструкция ONS ограничивает выполнение инструкции EVENT одним сканированием.

Инструкция EVENT запускает выполнение задачи *Task_1* (событийная задача).

Task_1 (событийная задача)

Инструкция GSV устанавливает атрибут *Task_Status* (тег DINT) для данной событийной задачи. THIS в атрибуте Instance Name означает объект TASK для задачи, в которой находится данная инструкция (т.е. *Task_1*).



продолжение на следующей странице

Если *Task_Status.0* = 1, то инструкция EVENT запустила данную событийную задачу. В непрерывной задаче инструкция EVENT выполняется для первоначальной активизации регистрации.

Инструкция JMP вызывает переход контроллера к выполнению инструкции *Arm* LBL. При этом пропускается вся логика данной процедуры, кроме цепи, активизирующей регистрацию для данной оси.

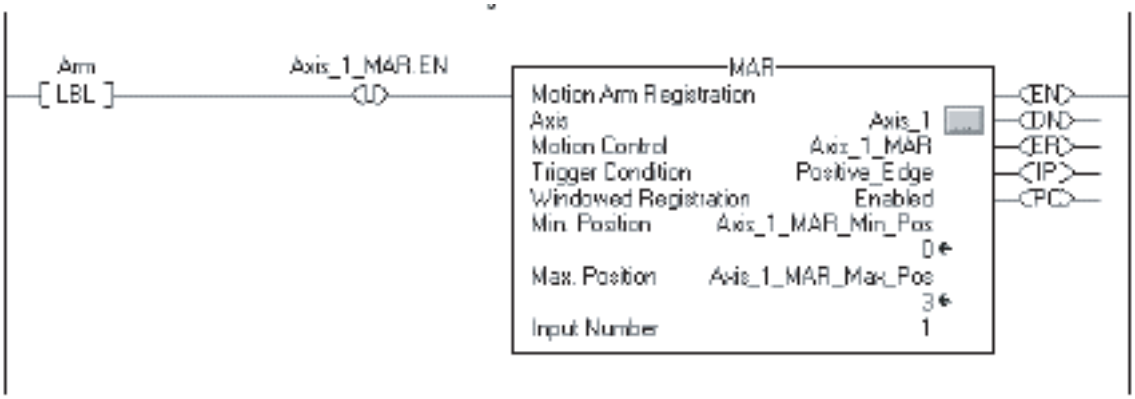
-
- Другая логика
-

Инструкция MAR выполняется при каждом выполнении данной задачи и активизирует регистрацию для оси *Axis_1*.

Инструкция OUT устанавливает бит EN инструкции MAR = 0.

- Инструкция MAR является инструкцией перехода.
- Для выполнения инструкции MAR входное состояние цепи для нее должно перейти из «ложно» в «истинно».
- При первом сбросе бита EN инструкция реагирует таким же образом, как если бы входное состояние цепи изменилось с «ложно» на «истинно».

Инструкция MAR активизирует данную ось для регистрации.

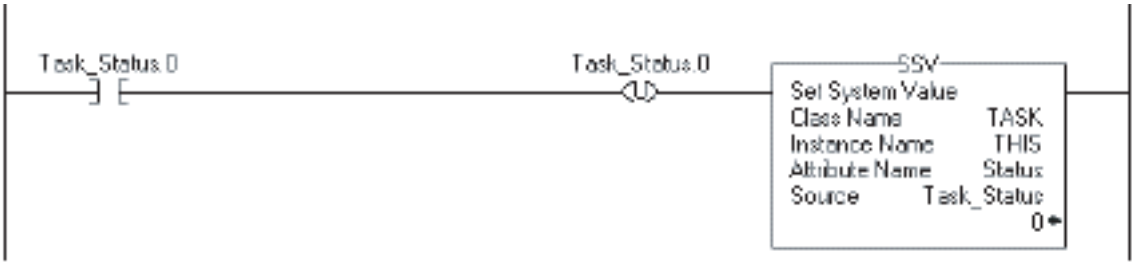


Контроллер не сбрасывает установленные биты атрибута Status. Чтобы использовать бит для новой информации о состоянии, вы должны вручную сбросить соответствующий бит.

Если *Task_Status.0* = 1, то сбросьте этот бит.

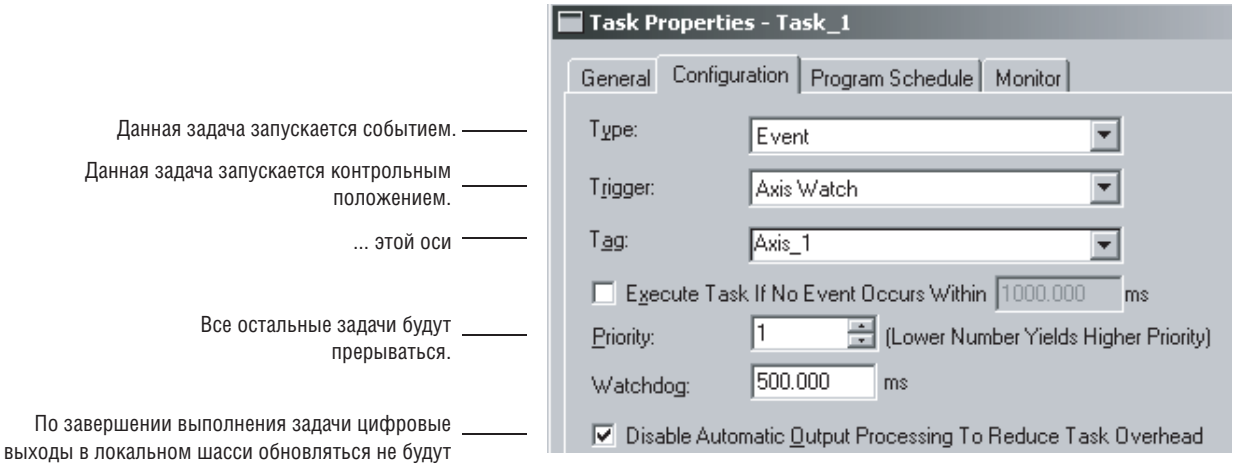
Инструкция OUT устанавливает *Task_Status.0* = 0.

Инструкция SSV устанавливает атрибут Status задачи THIS (*Task_1) = Task_Status*. Сюда входит и сброшенный бит.



Использование контроля оси в качестве триггера

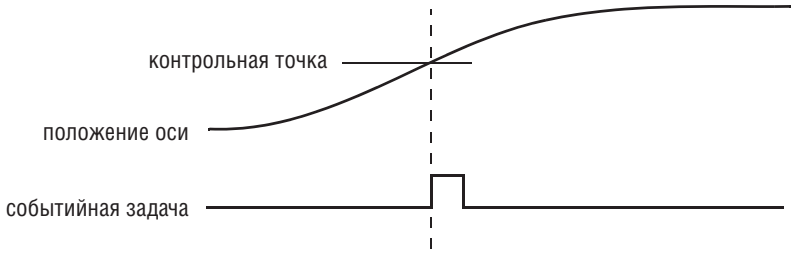
Чтобы событийная задача запускалась контрольным положением оси, используйте триггер *Axis Watch* (Контроль оси).



Данная событийная задача будет запущена по достижении указанной осью положения, заданного в качестве контрольного положения.

- Сначала вы должны активизировать ось для контрольного положения при помощи инструкции Motion Arm Watch (MAW).
- Операнд Trigger Condition инструкции MAW задает направление, в котором должна перемещаться ось, чтобы была запущена данная событийная задача.
- После достижения осью контрольного положения и запуска событийной задачи вы должны вновь активизировать ось для следующего контрольного положения.

Следующая временная диаграмма показывает взаимоотношение между контрольным положением и событийной задачей.



Контрольный перечень для задачи, запускаемой от контрольного положения оси

Для этого параметра:	Обеспечьте выполнение следующих требований:	
❑ 1. Контрольное положение	Используйте инструкцию MAR для задания контрольного положения. Это позволяет оси запускать событийную задачу по достижении ею заданного контрольного положения. • Сначала активизируйте ось для обнаружения первого контрольного положения. • После достижения осью контрольного положения и запуска событийной задачи вновь активизируйте ось для следующего контрольного положения • Реактивизируйте ось достаточно быстро для обнаружения каждого контрольного положения.	
	Если ваша обычная логика:	**То:**
	работает достаточно быстро для повторной активизации оси между контрольными положениями Например, ваша обычная логика всегда выполняет как минимум 2 сканирования между контрольными положениями.	При желании вы можете активизировать ось в вашей обычной логике.
	работает *недостаточно* быстро для повторной активизации оси	Активизируйте ось в соответствующей событийной задаче.
❑ 2. Приоритетность задачи	Назначьте данной событийной задаче наивысший приоритет. Если какая-либо периодическая задача имеет более высокий приоритет, событийная задача может быть вынуждена ожидать завершения выполнения такой периодической задачи.	
❑ 3. Количество событийных задач	Ограничивайте количество событийных задач. Каждая дополнительная задача уменьшает время обработки, отводимое для других задач. Это может привести к перекрытию.	
❑ 4. Автоматическая обработка вывода	Для событийной задачи вы, как правило, можете отключить автоматическую обработку вывода (настройка по умолчанию). Это уменьшает истекшее время выполнения задачи. Для принятия решения по этой настройке воспользуйтесь рисунком 4.1 на странице 4-14.	

ПРИМЕР

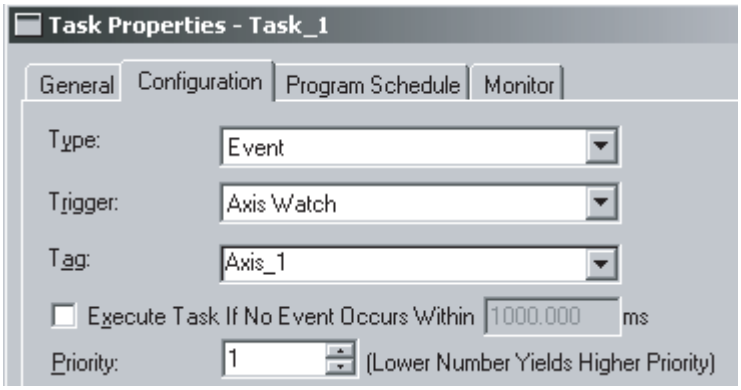
На участке наклейки этикеток линии разлива вы хотите проверять положение этикетки на бутылке.

- По достижении осью положения, заданного в качестве контрольной точки, осуществляется проверка этикетки и при необходимости выполняется корректировка.
- Из-за скорости линии вы должны активизировать ось для контрольного положения в событийной задаче.

Данная событийная задача запускается контрольным положением...

... для оси с именем Axis_1.

Данная событийная задача прерывает все остальные задачи.



Следующая логика активизирует и реактивизирует указанную ось для заданного контрольного положения.

Непрерывная задача

Если *Arm_Watch* = 1 (система готова к установке контрольного положения), то

Инструкция ONS ограничивает выполнение инструкции EVENT одним сканированием.

Инструкция EVENT запускает выполнение задачи *Task_1* (событийная задача).

Task_1 (событийная задача)

Инструкция GSV устанавливает атрибут *Task_Status* (тег DINT) для данной событийной задачи. THIS в атрибуте Instance Name означает объект TASK для задачи, в которой находится данная инструкция (т.е. *Task_1*).

продолжение на следующей странице

Если *Task_Status.0* = 1, то инструкция EVENT запустила данную событийную задачу. В непрерывной задаче инструкция EVENT выполняется для первоначальной установки контрольного положения.

Инструкция JMP вызывает переход контроллера к выполнению инструкции *Arm* LBL. При этом пропускается вся логика данной процедуры, кроме цепи, активизирующей данную ось для заданного контрольного положения (инструкция MAW).

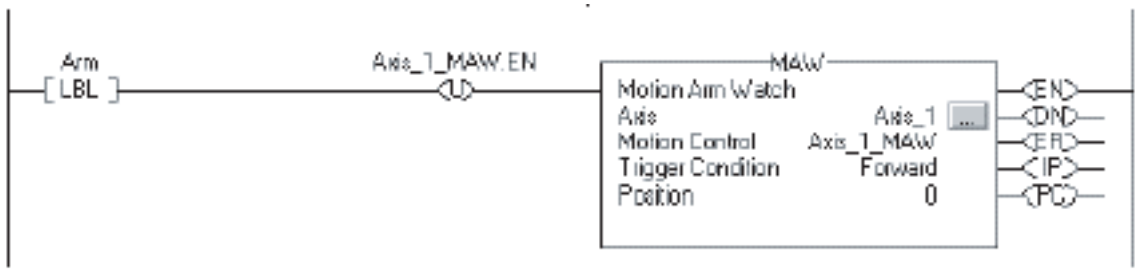
-
- Другая логика
-

Инструкция MAW выполняется при каждом выполнении данной задачи и активизирует ось *Axis_1* для заданного контрольного положения.

Инструкция OUT устанавливает бит EN инструкции MAW = 0.

- Инструкция MAW является инструкцией перехода.
- Для выполнения инструкции MAW входное состояние цепи для нее должно перейти из «ложно» в «истинно».
- При первом сбросе бита EN инструкция реагирует таким же образом, как если бы входное состояние цепи изменилось с «ложно» на «истинно».

Инструкция MAW активизирует данную ось для заданного контрольного положения.

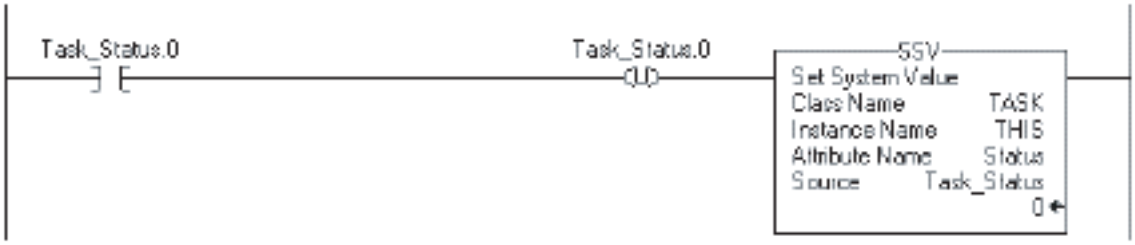


Контроллер не сбрасывает установленные биты атрибута Status. Чтобы использовать бит для новой информации о состоянии, вы должны вручную сбросить соответствующий бит.

Если *Task_Status.0* = 1, то сбросьте этот бит.

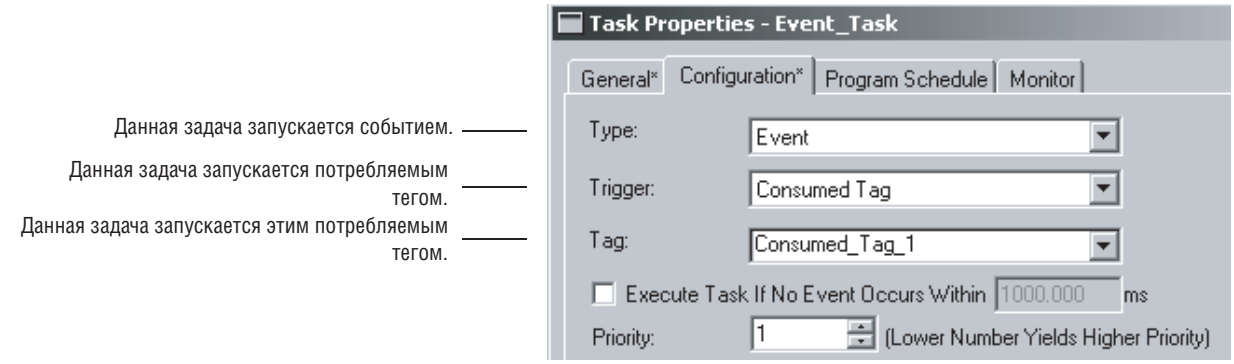
Инструкция OUT устанавливает *Task_Status.0* = 0.

Инструкция SSV устанавливает атрибут Status задачи THIS (*Task_1) = Task_Status*. Сюда входит и сброшенный бит.

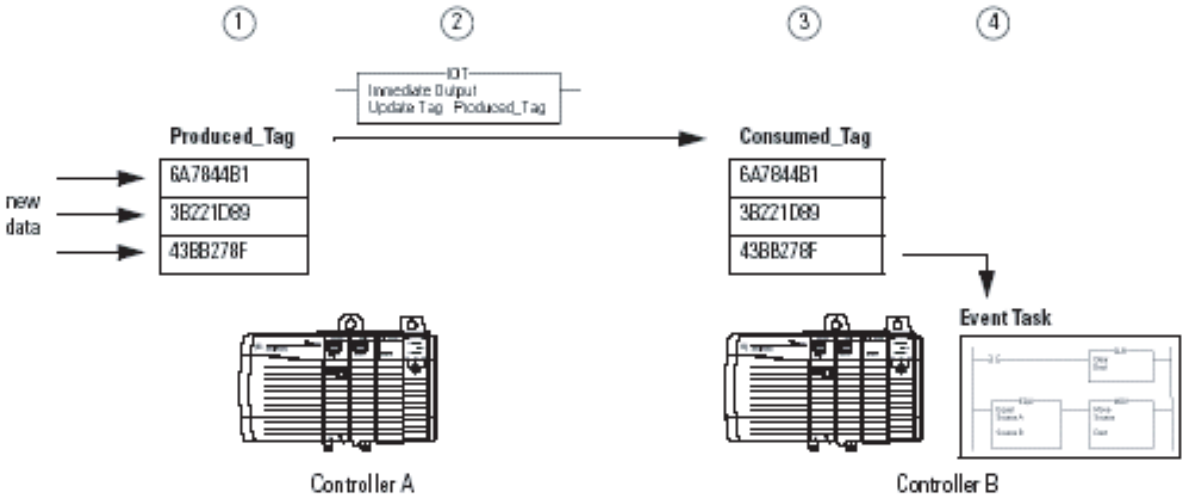


Использование потребляемого тега в качестве триггера

Чтобы событийная задача запускалась на основе данных потребляемого тега, используйте триггер *Consumed Tag* (Потребляемый тег).



Взаимоотношение между производимым и потребляемым тегом может обеспечивать отправку триггера событийной задачи, наряду с данными, в потребляющий контроллер. Как правило, для отправки триггера событийной задачи в потребляющий контроллер вы используете инструкцию Immediate Output (IOT).



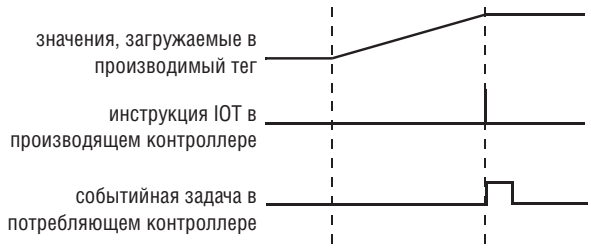
Описание:	
①	В Контроллере A логика обновляет значения производимого тега.
②	По завершении обновления Контроллер A выполняет инструкцию IOT для отправки соответствующих данных и триггера событийной задачи в Контроллер B.
③	Контроллер B потребляет новые данные.
④	После того, как контроллер B обновит потребляемый тег, он выполняет событийную задачу.

Тип сети, соединяющей контроллеры, определяет время получения потребляющим контроллером новых данных и триггера событийной задачи посредством инструкции IOT.

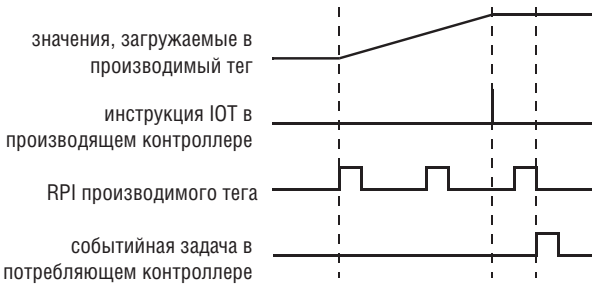
При использовании этого контроллера:	В этой сети:	Потребляющее устройство получает данные и триггер событийной задачи:
ControlLogix	объединительная плата	немедленно
	сеть EtherNet/IP	немедленно
	сеть ControlNet	через фактический межпакетный интервал (API) для потребляемого тега (соединения)
SoftLogix5800	Вы можете производить и потреблять теги только в сети ControlNet	через фактический межпакетный интервал (API) для потребляемого тега (соединения)

Следующие диаграммы позволяют сравнить получение данных при помощи инструкции IOT в сетях EtherNet/IP и ControlNet.

Сеть EtherNet/IP (контроллер ControlLogix)

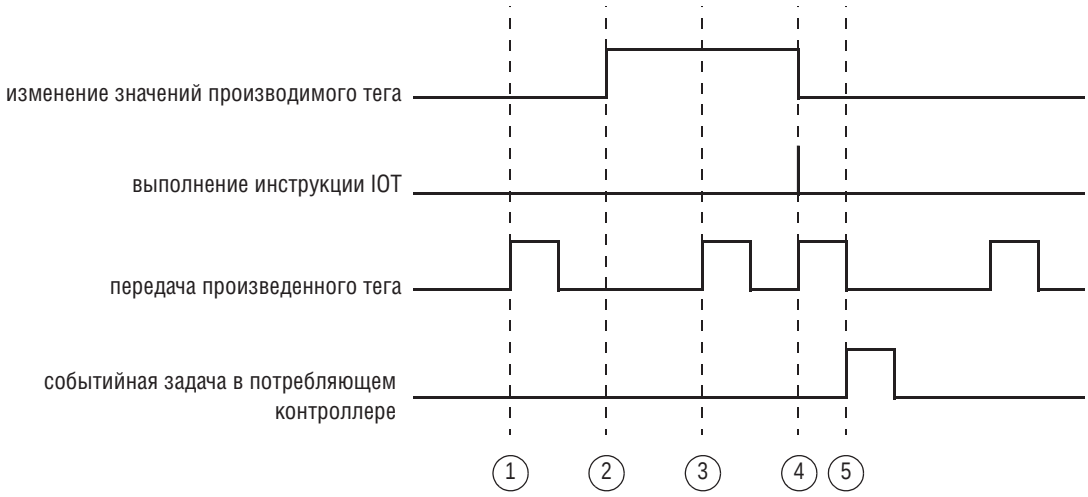


Сеть ControlNet



Поддержание целостности данных

Событийная задача с триггером в виде потребляемого тега обеспечивает простой механизм передачи данных контроллеру и гарантии того, что контроллер не будет использовать данные во время их изменения.

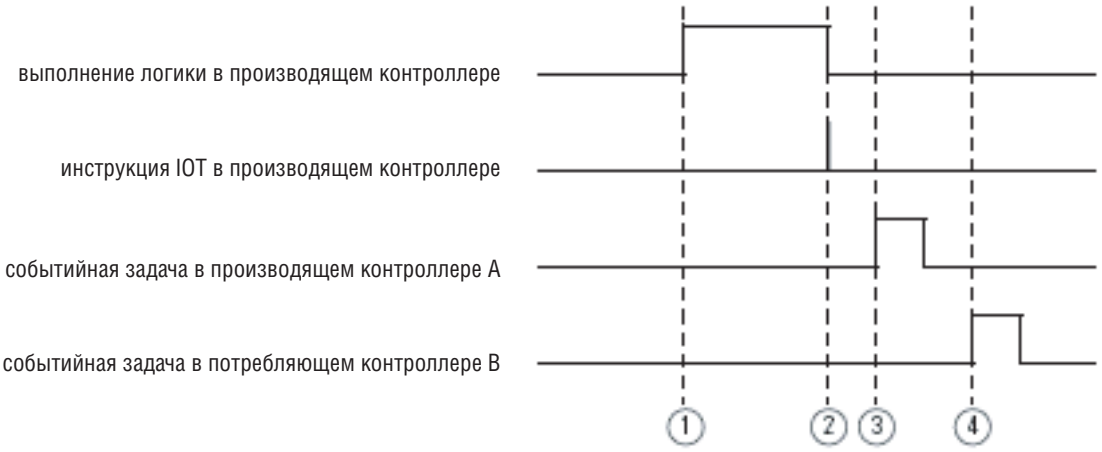


Описание:	
①	Для производимого тега наступает RPI. Производимый тег передает старые данные потребляющему контроллеру.
②	Производящий контроллер начинает обновлять значения производимого тега
③	Для производимого тега вновь наступает RPI. Производимый тег передает смесь из старых и новых данных потребляющему контроллеру.
④	Производящий контроллер завершает обновление значений производимого тега. Производящий контроллер выполняет инструкцию Immediate Output (IOT). Производимый тег немедленно передает все новые данные потребляющему контроллеру.
⑤	По получении потребляющим контроллером всех данных он выполняет событийную задачу.

Хотя производящий контроллер выполняет инструкцию IOT сразу же после загрузки в него новых данных, событийная задача (в потребляющем контроллере) будет запущена лишь тогда, когда потребляющий контроллер получит все новые данные. Это обеспечивает работу контроллера с полным пакетом новых данных.

Синхронизация нескольких контроллеров

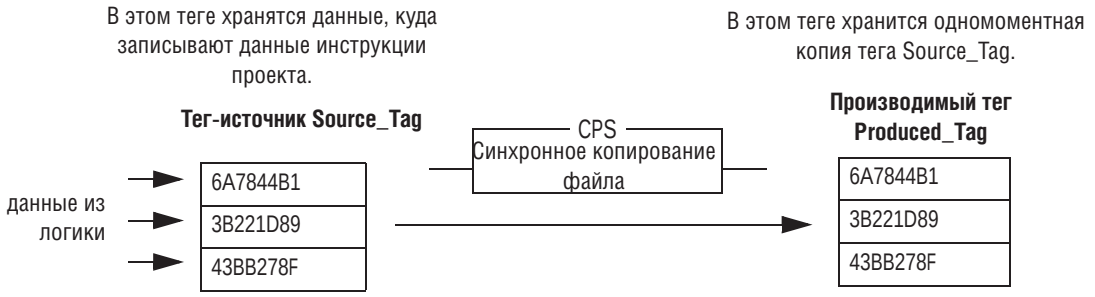
Также вы можете использовать взаимоотношение между производимым и потребляемым тегом для синхронизации контроллеров. В этом случае производимый/потребляемый тег служит лишь в качестве пускового механизма.



Описание:	
①	Первый контроллер выполняет действие, с которым должны синхронизироваться другие контроллеры.
②	По завершении выполнения этого действия контроллер выполняет инструкцию IOT. Эта инструкция использует производимый тег в качестве своего адресата.
③	По получении произведенного тега контроллером A он выполняет свою событийную задачу.
④	По получении произведенного тега контроллером B он выполняет свою событийную задачу.

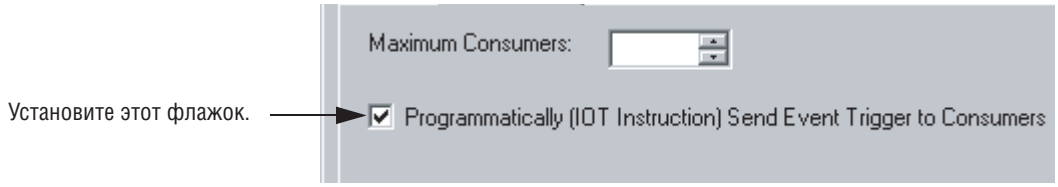
Контрольный перечень для производящего контроллера

Для этого	Проверьте следующее:
❑ 1. Буферизация данных	Если вы хотите одномоментно отправить полный образ данных, создайте копию этих данных как показано ниже:



Инструкция CPS не позволяет никаким операциям контроллера изменять данные в процессе копирования. Задачи, пытающиеся прервать инструкцию CPS, задерживаются до завершения копирования.

❑ 2. Свойства потребляемого тега	В свойствах соединения (Connection properties) производимого тега установите следующий флажок:

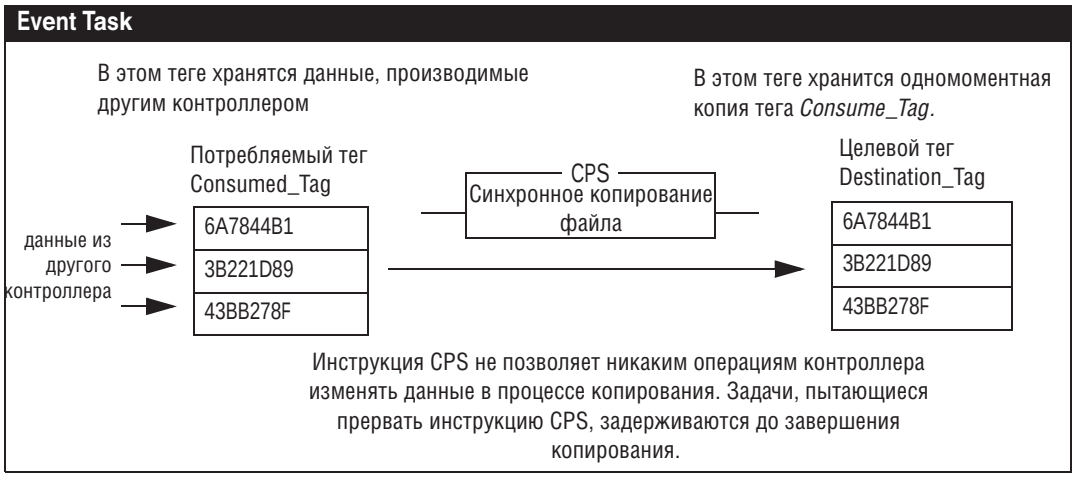


Если этот флажок не будет установлен, производящий контроллер будет запускать событийную задачу в конце всякой задачи, автоматически обновляющей локальные выходы. Иными словами, событие будет запускаться сканированием задачи наряду с инструкцией IOT

❑ 3. Инструкция IOT	Используйте инструкцию IOT в той точке вашей логике, где вы хотите запускать событийную задачу. Инструкция IOT отменяет RPI для данного тега и осуществляет немедленную передачу триггера событийной задачи и данных тега.

Контрольный перечень для потребляющего контроллера

Для этого:	Проверьте следующее:
❑ 1. Буферизация данных	Если вы хотите гарантировать, чтобы контроллер не использовал данные потребляемого тега в процессе изменения этих данных, используйте копию потребляемого тега. Для копирования данных используйте событийную задачу, как показано ниже:



❑ 2. Приоритетность задачи	Назначьте данной событийной задаче наивысший приоритет. Если какая-либо периодическая задача имеет более высокий приоритет, событийная задача может быть вынуждена ожидать завершения выполнения такой периодической задачи.
❑ 3. Количество событийных задач	Ограничивайте количество событийных задач. Каждая дополнительная задача уменьшает время обработки, отводимое для других задач. Это может привести к перекрытию.
❑ 4. Автоматическая обработка вывода	Для событийной задачи вы, как правило, можете отключить автоматическую обработку вывода (настройка по умолчанию). Это уменьшает истекшее время выполнения задачи. Для принятия решения по этой настройке воспользуйтесь рисунком 4.1 на странице 4-14.

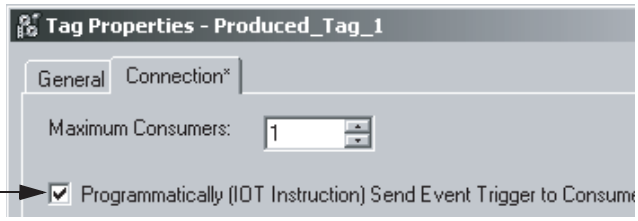
ПРИМЕР

При продвижении детали по производственной линии каждой станции требуются технические условия на данную деталь, относящиеся к конкретной станции. Чтобы исключить возможность действий со стороны станции на основе старых данных, событийная задача сигнализирует о поступлении новых данных для очередной детали.

Производящий контроллер

Этот контроллер управляет станцией 24 и производит данные для следующей станции (станция 25). Чтобы сигнализировать о передаче новых данных, контроллер использует следующие элементы:

Свойства производимого тега



Производимый тег Produced_Tag настроен на обновление своего триггера событийной задачи посредством инструкции IOT.

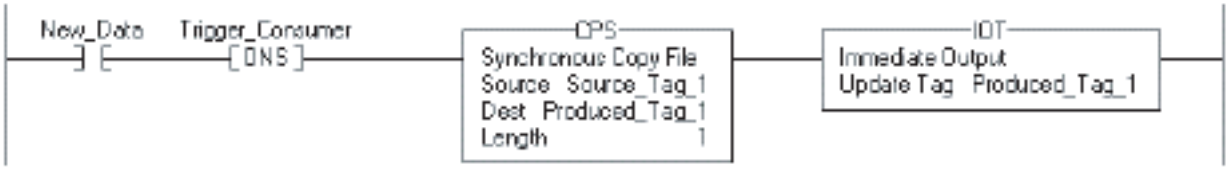
Производимый тег Produced_Tag настроен на обновление своего триггера событийной задачи посредством инструкции IOT.

Релейная логика

Если *New_Data* = on, то для одного сканирования происходит следующее:

Инструкция CPS устанавливает производимый тег равным тегу-источнику: *Produced_Tag_1= Source_Tag_1*.

Инструкция IOT обновляет *Produced_Tag_1* и отправляет обновленные данные в потребляющий контроллер (станция 25). По получении потребляющим контроллером этого обновления он запускает соответствующую событийную задачу.



продолжение на следующей странице

Потребляющий контроллер

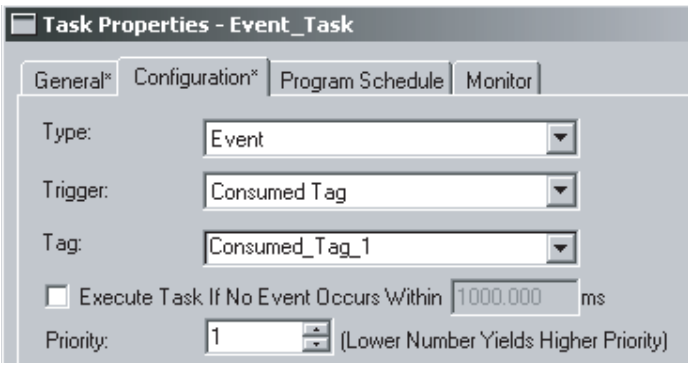
Контроллер на станции 25 использует данные, произведенные станцией 24. Для определения поступления новых данных этот контроллер использует событийную задачу.

Свойства событийной задачи

Данная задача запускается событием.

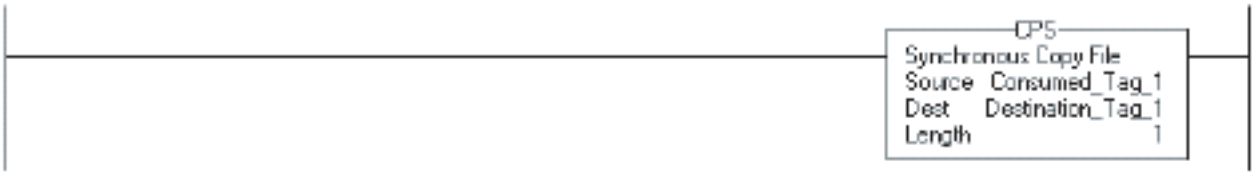
Данная задача запускается потребляемым тегом.

Данная задача запускается этим потребляемым тегом.



Релейная логика в событийной задаче

При выполнении событийной задачи инструкция CPS устанавливает тег-приемник равным потребляемому тегу: *Destination_Tag_1= Consumed_Tag_1* (значения, полученные от производящего контроллера). Остальная логика в этом контроллере использует значения из *Destination_Tag_1*.



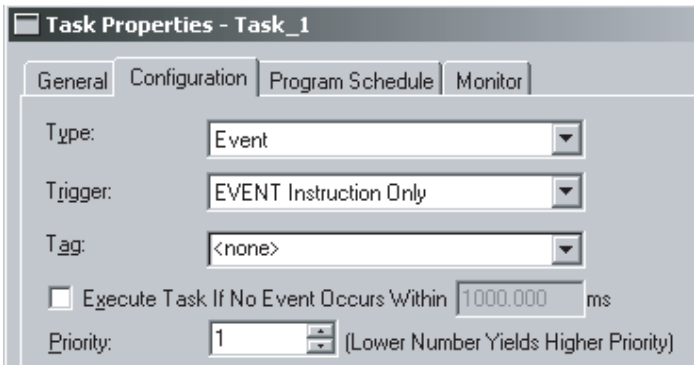
Использование инструкции EVENT в качестве триггера

Чтобы запускать событийную задачу на основе условий вашей логики, используйте триггер *EVENT Instruction Only* (Только инструкция EVENT).

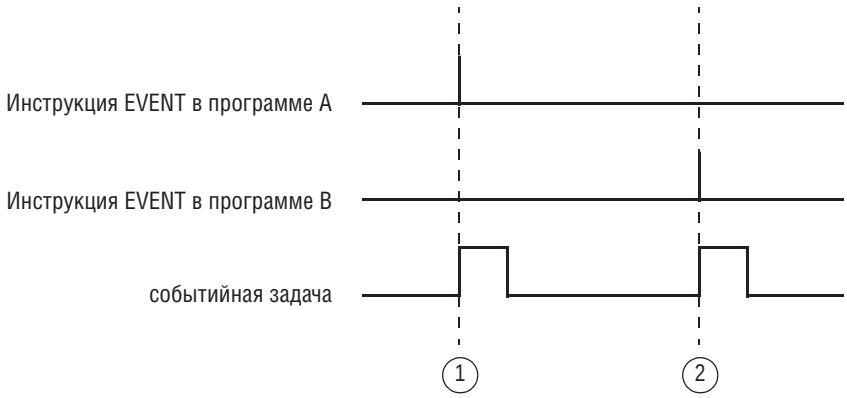
Данная задача запускается событием

Данная задача запускается инструкцией EVENT

Никакого тега не требуется.



Триггер *EVENT Instruction Only* требует использования инструкции Trigger Event Task (EVENT) для запуска задачи. Вы можете использовать инструкцию EVENT в нескольких точках вашего проекта. При каждом выполнении этой инструкции она будет запускать указанную событийную задачу.



Описание:	
①	Программа A выполняет инструкцию EVENT. Событийная задача, заданная данной инструкцией EVENT, выполняется один раз.
②	Программа B выполняет инструкцию EVENT. Событийная задача, заданная данной инструкцией EVENT, выполняется один раз.

Программное определение того, запустила ли инструкция EVENT задачу

Чтобы определить, запустила ли инструкция EVENT событийную задачу, используйте инструкция Get System Value (GSV) для проверки атрибута Status для соответствующей задачи.

Таблица 4.3 Атрибут Status объекта TASK

Атрибут:	Тип данных:	Инструкция:	Описание:	
Status	DINT	GSV	Предоставляет информацию о состоянии задачи. После установки бита контроллером вы должны вручную сбросить его для обнаружения возникновения другой ошибки такого типа.	
		SSV	**Чтобы определить, имеет ли место следующее:**	**Проверьте этот бит:**
			Инструкция EVENT запустила задачу (только для событийной задачи).	0
			Тайм-аут запустил задачу (только для событийной задачи).	1
			Для данной задачи произошло перекрытие.	2

Контроллер не сбрасывает установленные биты атрибута Status.

- Чтобы использовать бит для новой информации о состоянии, вы должны вручную сбросить этот бит.
- Используйте инструкцию Set System Value (SSV) для установки этого атрибута на другое значение.

Контрольный перечень для задачи, запускаемой инструкцией EVENT

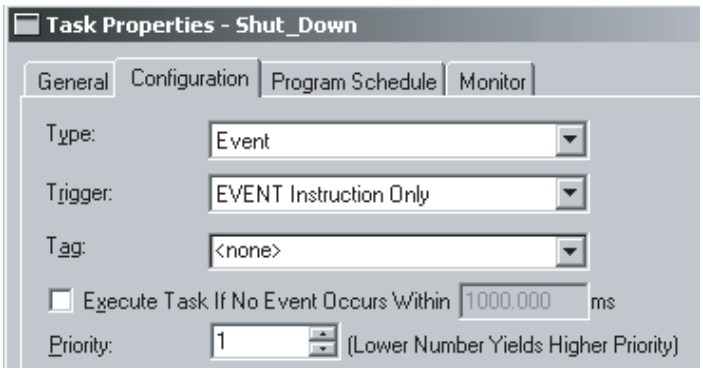
	Для этого:	Обеспечьте следующее:
❑	1. Инструкция EVENT	Используйте инструкцию Trigger Event Task (EVENT) в каждой точке вашей логики, где вы хотите запускать событийную задачу.
❑	2. Приоритетность задачи	Назначьте данной событийной задаче наивысший приоритет. Если какая-либо периодическая задача имеет более высокий приоритет, событийная задача может быть вынуждена ожидать завершения выполнения такой периодической задачи.
❑	3. Количество событийных задач	Ограничивайте количество событийных задач. Каждая дополнительная задача уменьшает время обработки, отводимое для других задач. Это может привести к перекрытию.
❑	4. Автоматическая обработка вывода	Для событийной задачи вы, как правило, можете отключить автоматическую обработку вывода (настройка по умолчанию). Это уменьшает истекшее время выполнения задачи. Для принятия решения по этой настройке воспользуйтесь рисунком 4.1 на странице 4-14.

ПРИМЕР

Контроллер использует несколько программ, но общую процедуру останова. Каждая из программ использует тег с именем *Shut_Down_Line* в области видимости программы, который включается при обнаружении программой условия, требующего останова.

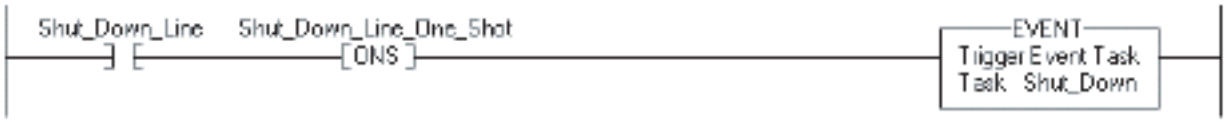
Свойства событийной задачи

Данная задача запускается событием. →
Данная задача запускается инструкцией EVENT. →
Никакого тега не требуется. →
Все остальные задачи прерываются. →



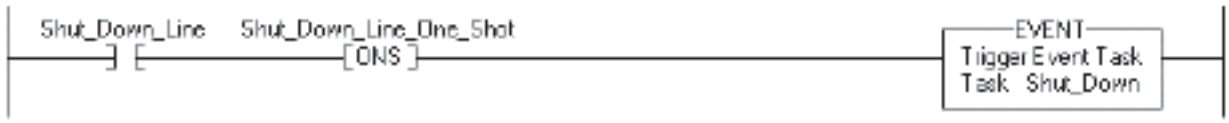
Релейная логика в программе Program_A

Если *Shut_Down_Line* = on (условия требуют останова), то
Задача *Shut_Down* выполняется один раз



Релейная логика в программе Program_B

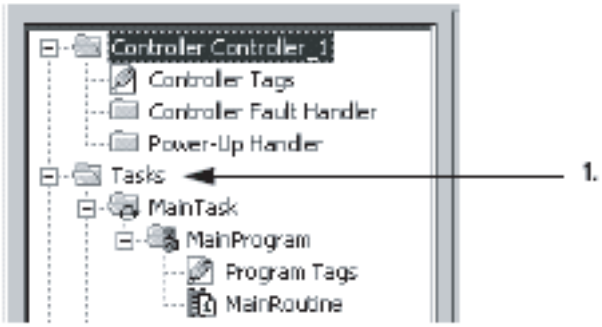
Если *Shut_Down_Line* = on (условия требуют останова), то
Задача *Shut_Down* выполняется один раз



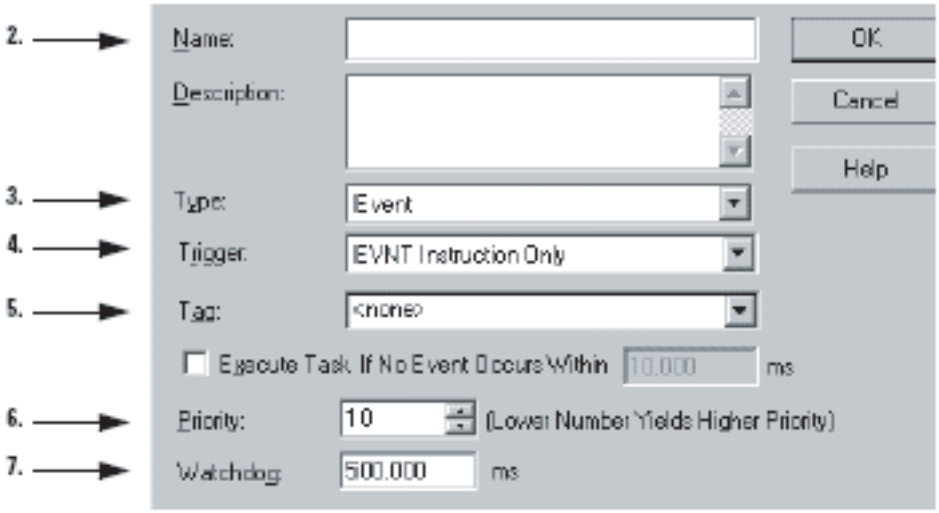
Создание задачи

Создание событийной задачи

Для создания событийной задачи:



1. В организаторе контроллера щелкните правой кнопкой мыши по папке *Tasks* (Задачи) и выберите *New Task* (Новая задача)..



2. В текстовом поле *Name* (Имя) введите **имя** данной задачи

3. В списке *Type* (Тип) выберите *Event* (Событие).

4. В списке *Trigger* (Триггер) выберите триггер для данной задачи.

5. В списке *Tag* (Тег) выберите тег, содержащий запускающие данные.

6. В текстовом поле *Priority* (Приоритет) введите **приоритет** данной задачи.

7. В текстовом поле *Watchdog* (Сторожевой таймер) введите время **сторожевого таймера** для данной задачи.

8. Нажмите 

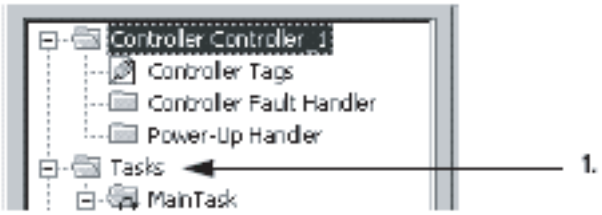
Создание периодической задачи

Периодическая задача выполняет определенную функцию или функции с заданной частотой.

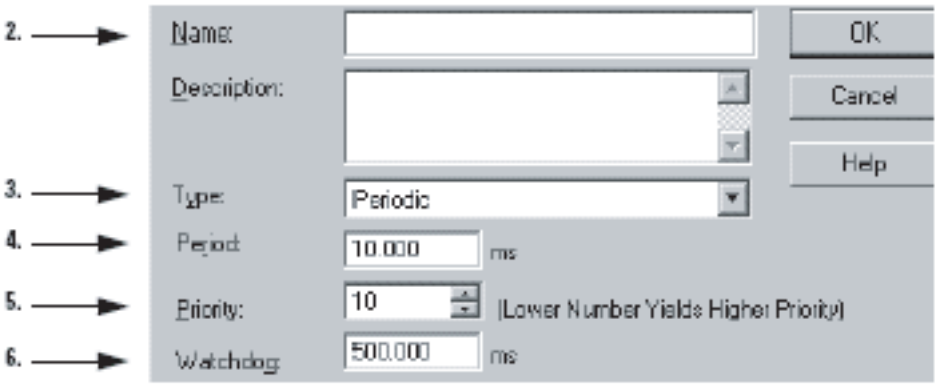
ВАЖНО

Временной период задачи должен быть больше суммы времен выполнения всех программ этой задачи.

- Если контроллер обнаружит появление триггера периодической задачи для задачи, которая уже работает, то возникнет неосновная ошибка (перекрытие).
- Приоритеты и времена выполнения других задач также могут приводить к перекрытию.



1. В организаторе контроллера щелкните правой кнопкой мыши по папке *Tasks* (Задачи) и выберите *New Task* (Новая задача).



2. В текстовом поле *Name* (Имя) введите **имя** данной задачи.

3. В списке *Type* (Тип) выберите *Periodic* (Периодическая) (настройка по умолчанию).

4. В текстовом поле *Period* (Период) введите период, с которым должна выполняться данная задача.

5. В текстовом поле *Priority* (Приоритет) введите **приоритет** данной задачи.

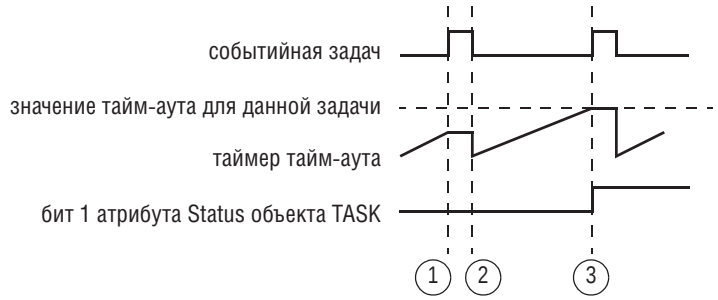
6. В текстовом поле *Watchdog* (Сторожевой таймер) введите время **сторожевого таймера** для данной задачи.

7. Нажмите 

Задание значения тайм-аута для событийной задачи

Если вы хотите, чтобы ваша событийная задача автоматически выполнялась, если соответствующий триггер не появится в течение определенного времени, то назначьте для этой задачи значение тайм-аута.

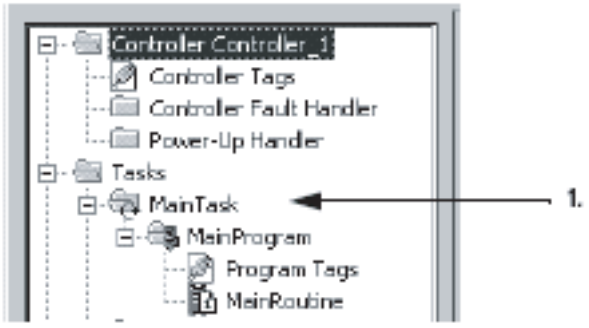
После выполнения событийной задачи значение ее таймера тайм-аута начинает дискретно увеличиваться. Если таймер достигнет заданного для него значения до запуска соответствующей событийной задачи, эта событийная задача будет автоматически выполняться.



Описание:	
①	Событийная задача выполняется Время тайм-аута прекращает увеличиваться.
②	Событийная задача выполнена Происходит сброс таймера тайм-аута, и его время начинает увеличиваться.
③	Таймер тайм-аута достигает значения тайм-аута. Событийная задача автоматически выполняется. Устанавливается бит 1 в атрибуте Status объекта TASK.

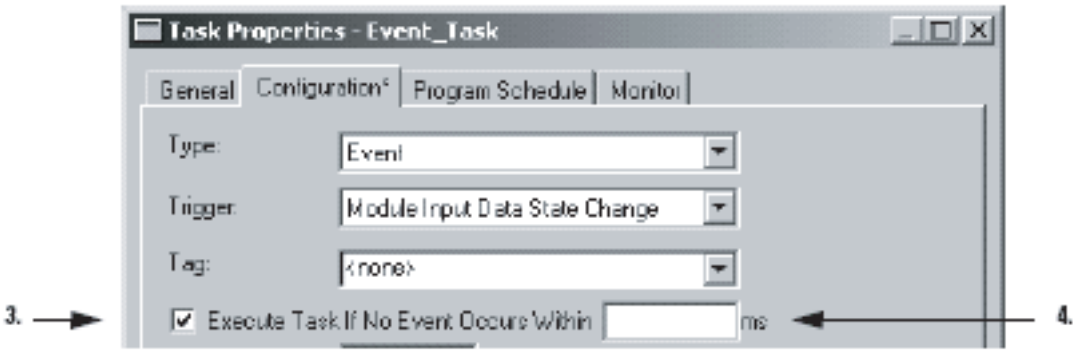
Задание значения тайм-аута для событийной задачи

Чтобы задать значение тайм-аута для событийной задачи:



1. В организаторе контроллера щелкните правой кнопкой мыши по событийной задаче и выберите *Properties* (Свойства).

2. Щелкните по закладке *Configuration* (Конфигурация).



3. Установите флажок *Execute Task If No Event Occurs Within* (Выполнять задачу, если событие не возникает в течение) .

4.. Введите время тайм-аута в миллисекундах.

5.. Нажмите OK

Программное конфигурирование тайм-аута

Чтобы программно сконфигурировать тайм-аут, используйте инструкцию Get System Value (GSV) для обращения к следующим атрибутам соответствующей задачи.

Таблица 4.4 Атрибут Status объекта TASK

Атрибут:	Тип данных:	Инструкция:	Описание:	
Rate	DINT	GSV	**Если тип задачи:**	**То атрибут Rate определяет:**
		SSV	периодическая	Период для данной задачи. Время указывается в микросекундах.
			событийная	Значение тайм-аута для данной задачи. Время указывается в микросекундах.
EnableTimeOut	DINT	GSV	Разрешает или запрещает функцию тайм-аута для событийной задачи.	
		SSV	**Чтобы:**	**Установите этот атрибут на:**
			запретить функцию тайм-аута	0 (по умолчанию)
			разрешить функцию тайм-аута	1 (или любое ненулевое значение)

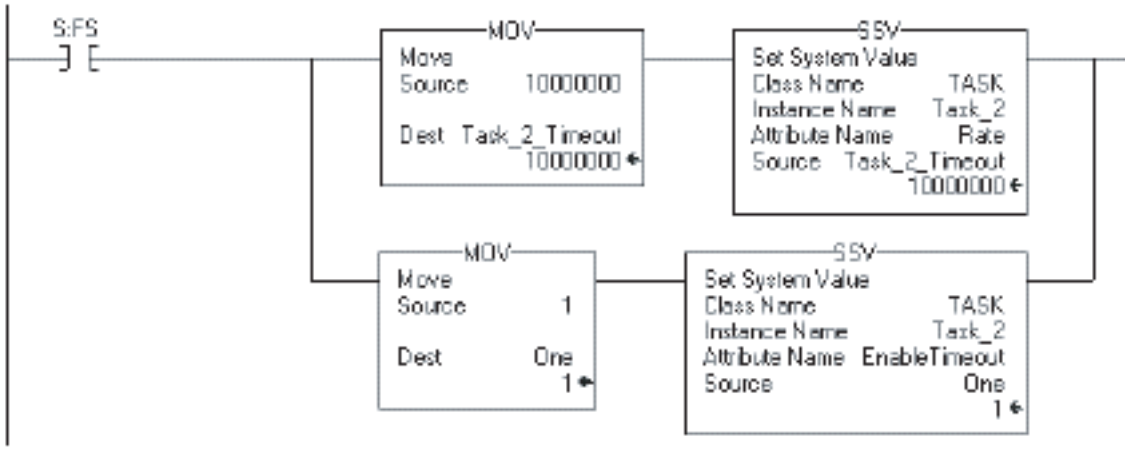
ПРИМЕР

Программное конфигурирование тайм-аута

Чтобы значение тайм-аута всегда было определено и разрешено для событийной задачи, следующая логика конфигурирует тайм-аут при переходе контроллера в режим выполнения.

Если S:FS = 1 (первое сканирование), то устанавливается значение тайм-аута для задачи *Task_2* и разрешается функция тайм-аута:

1. Первая инструкция MOV устанавливает *Task_2_Timeout* = 10000000 μc (значение DINT). Затем инструкция SSV устанавливает атрибут Rate для задачи *Task_2* = *Task_2_Timeout*. Это конфигурирует значение тайм-аута для данной задачи.
2. Вторая инструкция MOV устанавливает *One* = 1 (значение DINT). Затем инструкция SSV устанавливает атрибут EnableTimeout для задачи *Task_2* = *One*. Это разрешает функцию тайм-аута для данной задачи.



Программное определение того, произошел ли тайм-аут

Чтобы определить, выполняется ли событийная задача по тайм-ауту, используйте инструкцию Get System Value (GSV) для проверки атрибута Status данной задачи.

Таблица 4.5 Атрибут Status объекта TASK

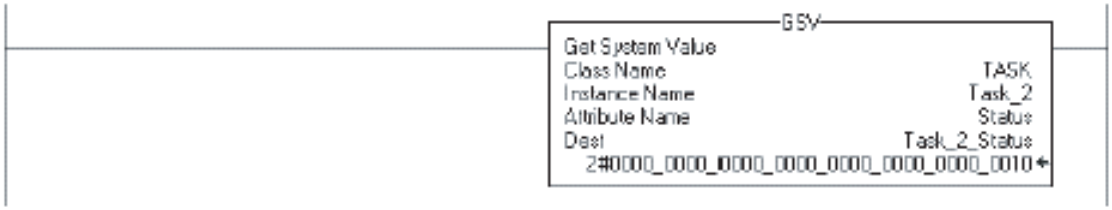
Атрибут:	Тип данных:	Инструкция:	Описание:	
Status	DINT	GSV	Предоставляет информацию о состоянии задачи. После установки бита контроллером вы должны вручную сбросить его для обнаружения возникновения другой ошибки такого типа.	
		SSV	**Чтобы определить, имеет ли место следующее:**	**Проверьте этот бит:**
			Инструкция EVENT запустила задачу (только для событийной задачи).	0
			Тайм-аут запустил задачу (только для событийной задачи).	1
			Для данной задачи произошло перекрытие.	2

ПРИМЕР

Задание значения тайм-аута для событийной задачи

Если для событийной задачи происходит тайм-аут, это может означать нарушение связи с запускающим устройством. Это требует остановки процесса. Для останова контроллера событийная задача вызывает процедуру ошибки для соответствующей программы и передает заданный пользователем код ошибки (в этом примере 999).

1. Инструкция GSV устанавливает *Task_2_Status* = атрибут Status для задачи *Task_2* (значение DINT).



2. Если *Task_2_Status.1* = 1, то произошел тайм-аут, поэтому контроллер останавливается и код основной ошибки устанавливается на 999:

Инструкция JSR вызывает процедуру ошибки для данной программы. Это порождает основную ошибку.

Код этой основной ошибки = 999 (значение входного параметра 999).

3. Если *Condition_1* = 1, то биты атрибута Status для задачи *Task_2* сбрасываются.

Инструкция SSV устанавливает атрибут Status задачи *Task_2* = *Zero*. Zero – это тег DINT, имеющий значение 0.

За дополнительной информацией по останову контроллера обращайтесь к разделу «Создание пользовательской основной ошибки» на стр. 15-13.

Для заметок:

Разработка последовательной функциональной схемы

Когда использовать эту процедуру

Используйте эту процедуру для разработки **последовательной функциональной схемы (ПФС)** для вашего процесса или системы. ПФС аналогична блок-схеме вашего процесса. Она определяет шаги или состояния, которые проходит ваша система. Используйте ПФС для:

- подготовки функциональной спецификации для вашей системы
- программирования и управления вашей системой в виде последовательности шагов и переходов

Использование ПФС для спецификации вашего процесса дает вам следующие преимущества:

- Поскольку ПФС является графическим представлением вашего процесса, она легче организуется и воспринимается, чем текстовый вариант. Кроме того, программное обеспечение RSLogix5000 позволяет вам:
  - добавлять примечания для пояснения шагов или выделения важной информации для последующего использования
  - распечатывать ПФС для предоставления этой информации другим лицам
- Поскольку контроллеры Logix5000 поддерживают ПФС, вам не придется повторно вводить спецификацию. Вы программируете свою систему одновременно с подготовкой ее спецификации.

Использование ПФС для программирования вашего процесса дает вам следующие преимущества:

- графическое разделение процессов на основные логические части (шаги)
- более быстрое повторрное выполнение отдельных частей вашей логики
- более простое представление информации на экране
- меньше времени на разработку и отладку вашей программы
- более быстрая и простая диагностика
- непосредственный доступ к точке логики, где произошла машинная ошибка
- простота обновления и модернизации

Как использовать эту процедуру

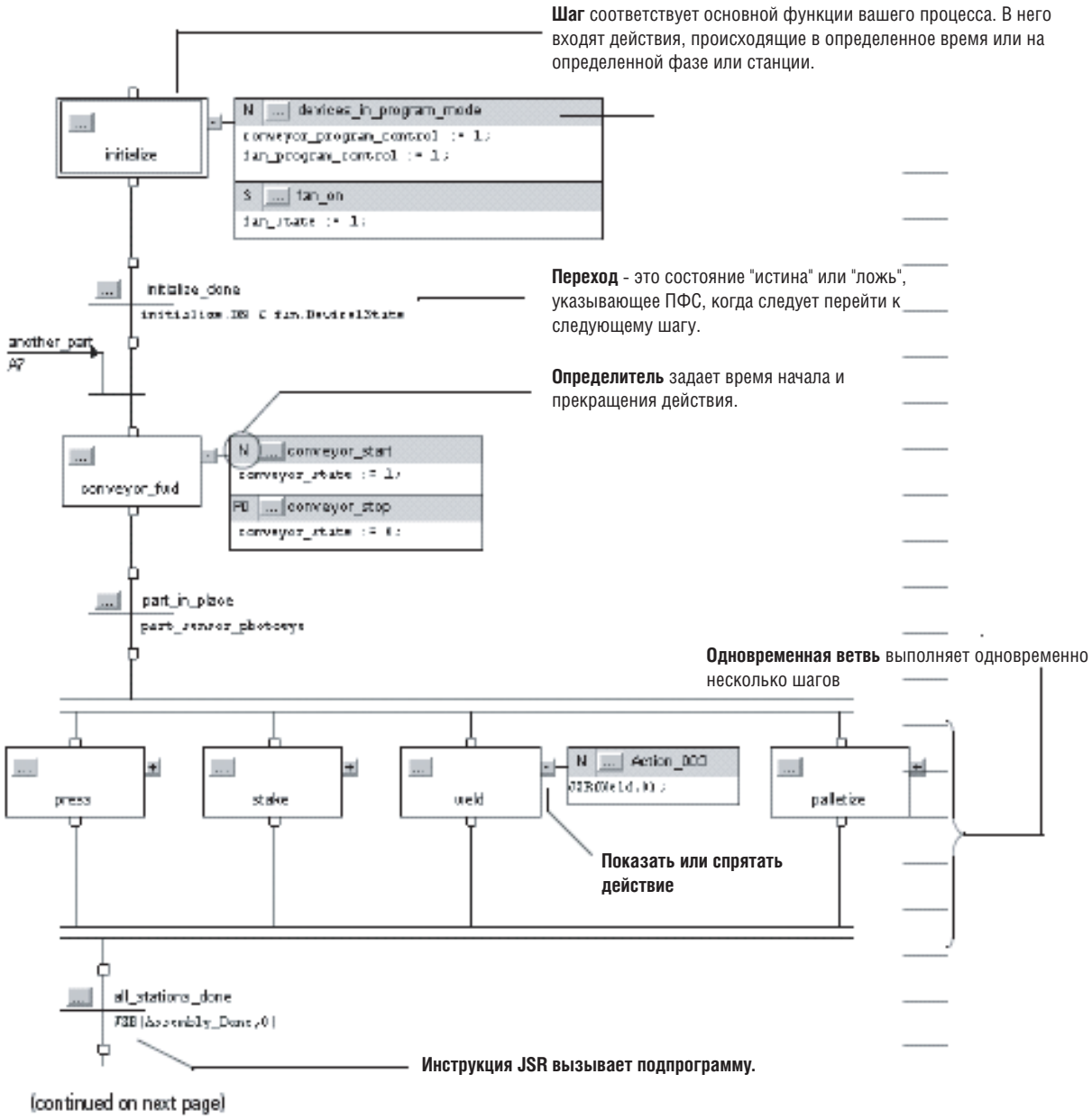
Как правило, разработка ПФС представляет собой итерационный процесс. По желанию вы можете использовать программное обеспечение RSLogix5000 для проектирования и доработки ПФС. Конкретные процедуры ввода ПФС приводятся в разделе «Программирование последовательной функциональной схемы» на стр. 6-1.

Что такое последовательная функциональная схема?

Последовательная функциональная схема (ПФС) аналогична блок-схеме. В ней используются шаги и переходы для выполнения конкретных операций или действий.

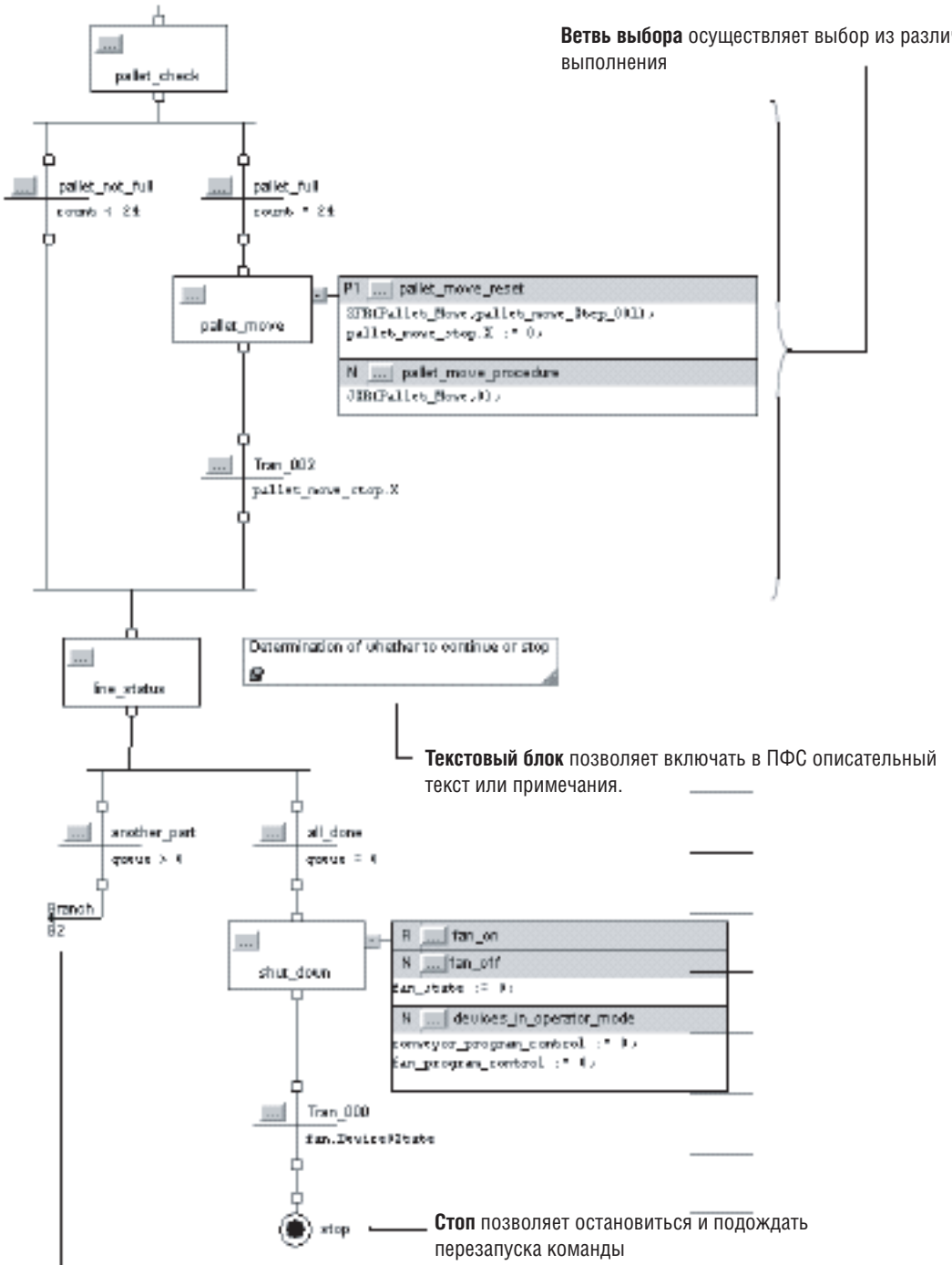
Пример ПФС, показывающий ее отдельные элементы, приводится на рисунках 5.1 и 5.2.

Рисунок 5.1 Пример ПФС



(продолжение на следующей странице)

Рисунок 5.2 Пример ПФС (продолжение с предыдущей страницы)



Как разработать ПФС: Обзор

Чтобы разработать ПФС, необходимо выполнить следующее:

- Определить задачи
- Выбрать способ выполнения ПФС
- Определить шаги вашего процесса
- Организовать шаги
- Добавить действия для каждого шага
- Описать каждое действие при помощи псевдокода
- Выбрать для действия определитель
- Задать условия перехода
- Задать переход по истечении определенного времени
- Выключить устройство в конце шага
- Оставить что-то во включенном состоянии между шагами
- Задать окончание ПФС
- Организовать вложенные ПФС
- Задать, когда следует вернуться в OC/JSR
- Приостановить или сбросить ПФС

В последующих разделах этой главы подробно описываются процедуры выполнения каждого из вышеперечисленных этапов.

Определение задач

Первым этапом разработки ПФС является отделение конфигурирования и регулирования устройств от подаваемых на устройства команд. Контроллеры Logix5000 позволяют вам разделить ваш проект на одну **непрерывную задачу** и несколько **периодических задач** и **событийных задач**.

1. Организуйте ваш проект следующим образом:

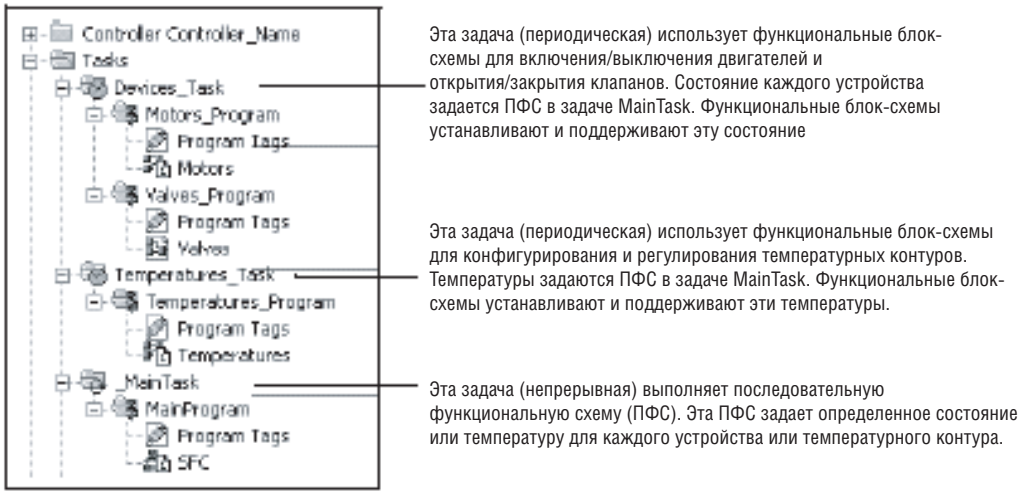
Эти функции:	Передаются:
конфигурирование и регулирование устройств	периодической задаче
управление переходом устройства в определенное состояние	ПФС в непрерывной задаче
установление последовательности выполнения вашего процесса	

2. Сгруппируйте функции, относящиеся к периодической задаче, по их частоте обновления. Создайте периодическую задачу для каждой частоты обновления.

   Например, для вашего двухпозиционного устройства может потребоваться более высокая частота обновления, чем для ваших контуров ПИД-управления. Используйте для них отдельные периодические задачи.

В следующем примере показан проект, использующий две периодические задачи для регулирования двигателей, клапанов и температурных контуров. Для управления процессом в этом проекте используется ПФС.

ПРИМЕР Определение задач



Выбор способа выполнения ПФС

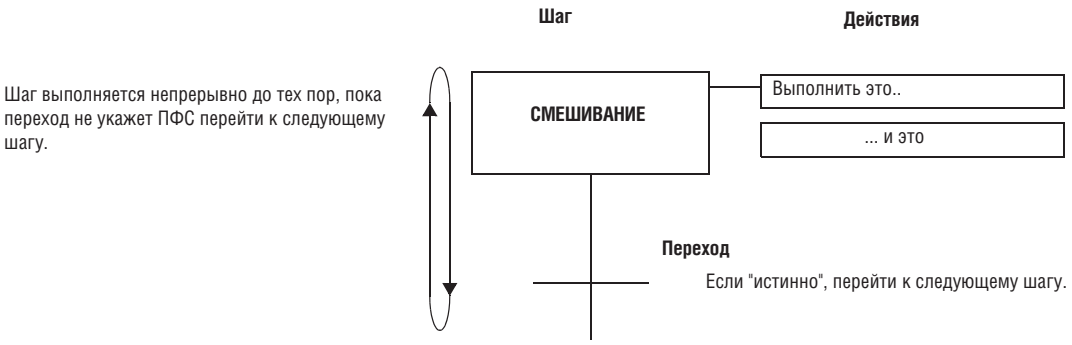
Для выполнения ПФС сконфигурируйте ее в качестве главной процедуры для программы или вызывайте ее как подпрограмму.

Если:	То:
ПФС является единственной процедурой в данной программе.	Сконфигурируйте ПФС в качестве главной процедуры для этой программы.
ПФС вызывает *все* остальные процедуры данной программы.	
Программе для выполнения требуются другие процедуры независимо от ПФС.	1. Сконфигурируйте другую процедуру в качестве главной процедуры для данной программы.
ПФС испольщует булевы действия.	2. Используйте главную процедуру для вызова ПФС как подпрограммы.

Если ПФС использует булевы действия, то другая логика должна выполняться независимо от ПФС и контролировать биты состояния ПФС.

Определение шагов процесса

Шаг соответствует основной функции вашего процесса. В него входят действия, происходящие в определенное время или на определенной фазе или станции.

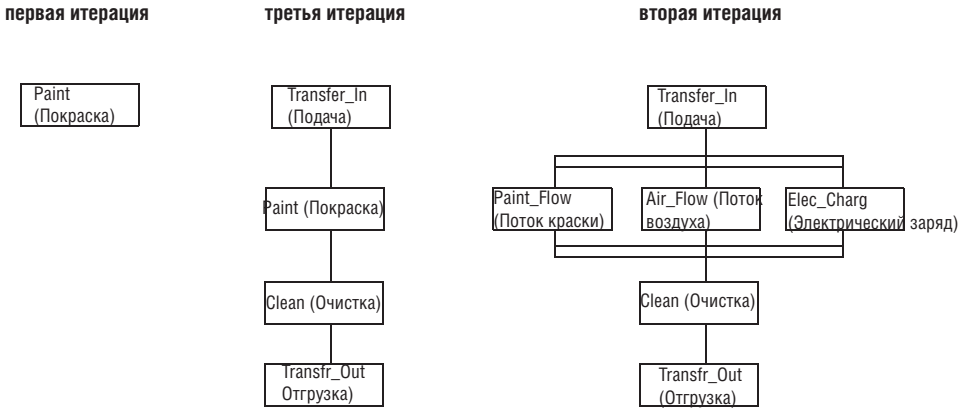


Переход заканчивает шаг. Переход определяет физические условия, которые должны иметь место или измениться, чтобы произошел переход к следующему шагу.

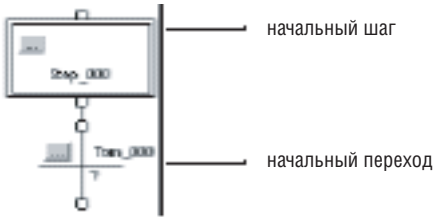
Руководящие указания

При определении шагов вашего процесса придерживайтесь следующих указаний:

- Начните с укрупненных шагов и уточняйте шаги за несколько итераций.



- При первом открытии процедуры ПФС она содержит начальный шаг и переход. Используйте этот шаг для инициализации вашего процесса.



- Для выделения шага определите физическое изменение в вашей системе, например, поступление в данную позицию новой детали, достижение определенной температуры или заданного времени, или выбор рецепта. Шаг – это действие, происходящее перед таким изменением.
- Остановите процесс определения шагов, когда вы получите осмысленные части. Например:

Такая организация шагов:	**Является:**
produce_solution	по-видимому, слишком укрупненной
set_mode, close_outlet, set_temperature, open_inlet_a, close_inlet_a, set_timer, reset_temperature, open_outlet, reset_mode	по-видимому, слишком мелкой
preset_tank, add_ingredient_a, cook, drain	по-видимому, практически правильной

Структура SFC_STEP

Каждый шаг использует тег для предоставления информации о данном шаге. К этой информации можно обратиться посредством либо диалогового окна *Step Properties* (Свойства шага), либо закладки *Monitor Tags* (Контроль тегов) в окне *Tags* (Теги):

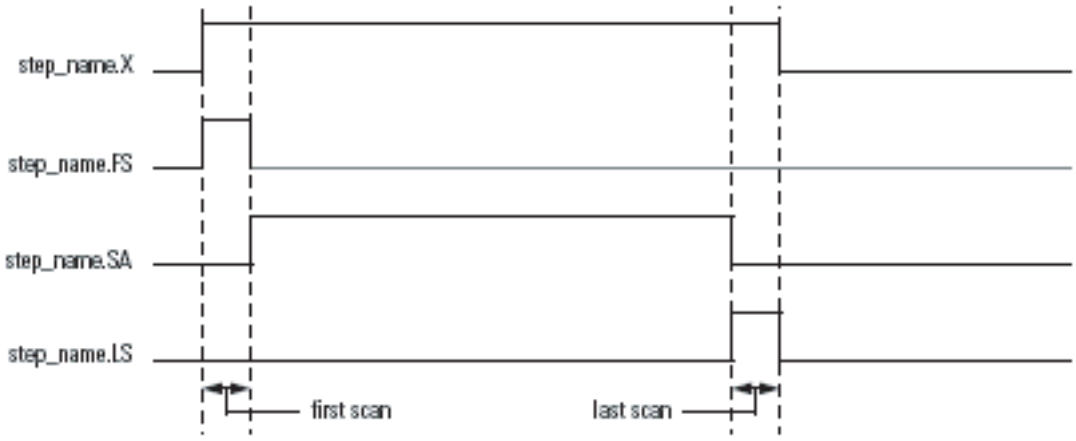
Если вы хотите:	То проверьте или установите этот член:	Тип данных:	Описание:
определить, сколько времени шаг был активен (в миллисекундах)	T	DINT	Когда шаг становится активным, значение Timer (T) сбрасывается и начинается прямой отсчет в миллисекундах. Таймер продолжает отсчет, пока данный шаг не станет неактивным, независимо от заданного значения Preset (PRE).
поставить флаг, когда активное состояние шага продлилось определенное время (в миллисекундах)	PRE	DINT	Введите время в член Preset (PRE). Когда Timer (T) достигнет этого заданного значения, бит выполнения Done (DN) установится и будет оставаться в установленном состоянии до того момента, когда данный шаг вновь станет активным. Также вы можете ввести численное выражение для вычисления времени во время выполнения.
	DN	BOOL	Когда Timer (T) достигнет значения Preset (PRE), бит выполнения Done (DN) установится и будет оставаться в установленном состоянии до того момента, когда данный шаг вновь станет активным.
поставить флаг, если шаг выполнялся недостаточно долго	LimitLow	DINT	Введите время в член LimitLow (в миллисекундах). • Если данный шаг станет неактивным до того, как таймер Timer (T) достигнет значения LimitLow, то установится бит AlarmLow. • Бит AlarmLow будет оставаться в установленном состоянии, пока вы его не сбросите. • Для использования этой функции сигнализации установите бит AlarmEnable (AlarmEn). Также вы можете ввести численное выражение для вычисления времени во время выполнения.
	AlarmEn	BOOL	Для использования битов сигнализации установите бит AlarmEnable (AlarmEn).
	AlarmLow	BOOL	Если данный шаг станет неактивным до того, как таймер Timer (T) достигнет значения LimitLow, то установится бит AlarmLow. • Этот бит будет оставаться в установленном состоянии, пока вы его не сбросите. • Для использования этой функции сигнализации установите бит AlarmEnable (AlarmEn).

Если вы хотите:	То проверьте или установите этот член:	Тип данных:	Описание:
поставить флаг, если шаг выполняется слишком долго	LimitHigh	DINT	Введите время в член LimitHigh (в миллисекундах). • Если Timer (T) достигнет значения LimtHigh, то установится бит AlarmHigh. • Бит AlarmHigh будет оставаться в установленном состоянии, пока вы его не сбросите. • Для использования этой функции сигнализации установите бит AlarmEnable (AlarmEn). Также вы можете ввести численное выражение для вычисления времени во время выполнения.
	AlarmEn	BOOL	Для использования битов сигнализации установите бит AlarmEnable (AlarmEn).
	AlarmHigh	BOOL	Если Timer (T) достигнет значения LimtHigh, то установится бит AlarmHigh. • Этот бит будет оставаться в установленном состоянии, пока вы его не сбросите. • Для использования этой функции сигнализации установите бит AlarmEnable (AlarmEn)
выполнить что-либо в то время, когда шаг активен (включая первое и последнее сканирование)	X	BOOL	Бит X установлен все время, пока шаг активен (выполняется). В общем случае мы рекомендуем использовать для этого указанное действие вместе с определителем *P1 Pulse (Rising Edge)* (Передний фронт импульса).
однократно выполнить что-либо в тот момент, когда шаг станет активным	FS	BOOL	Бит FS установлен во время первого сканирования шага. В общем случае мы рекомендуем использовать для этого указанное действие вместе с определителем *P1 Pulse (Rising Edge)*.
выполнить что-либо в то время, когда шаг активен, *за исключением* первого и последнего сканирования	SA	BOOL	Бит SA установлен все время, пока шаг активен, за исключением первого и последнего сканирования данного шага.
однократно выполнить что-либо во время последнего скнирования шага	LS	BOOL	Бит LS установлен все время последнего сканирования шага. Используйте этот бит только в том случае, если вы выполните следующее: В закладке *SFC Execution* диалогового окна C*ontroller Properties* (Свойства контроллера) установите опцию *Last Scan of Active Step* (Последнее сканирование активного шага) на *Don't Scan* (Не сканировать) или *Programmatic reset* (Программный сброс). В общем случае мы рекомендуем использовать для этого указанное действие вместе с определителем *P0 Pulse (Falling Edge)* (Задний фронт импульса)

Если вы хотите:	То проверьте или установите этот член:	Тип данных:	Описание:
определить цель инструкции SFC Reset (SFR)	Reset	BOOL	Инструкция SFC Reset (SFR) возвращает ПФС к шагу или стопу, указанному в этой инструкции: • Бит Reset указывает, к какому шагу или стопу вернется ПФС, чтобы вновь начать выполняться. • Сразу после выполнения SFC бит Reset сбрасывается.
определить максимальное время активного состояния шага в процессе любого из его выполнений	TMax	DINT	Используйте это для целей диагностики. Контроллер будет сбрасывать это значение лишь в том случае, если вы выберете *Restart at initail step* (Перезапуск с начального шага) в качестве *Restart Position* (Точки перезапуска), а контроллер перейдет в другой режим или его питание будет отключено с последующим включением.
определить, не принимает ли Timer (T) отрицательное значение	OV	BOOL	Используйте это для целей диагностики.
определить, сколько раз был активен какой-либо шаг	Count	DINT	Это *не* является подсчетом количества сканирований соответствующего шага. • Этот счетчик увеличивается на единицу каждый раз, когда соответствующий шаг становится активным. • Он вновь увеличивается на единицу лишь после того, как данный шаг станет неактивным, а затем вновь станет активным. • Счетчик сбрасывается лишь в том случае, если вы сконфигурируете перезапуск ПФС с начального шага. При такой настройке он сбрасывается при переходе контроллера из программного режима в режим выполнения.
использовать один тег для различных битов состояния данного шага	Status	DINT	

Для этого члена:	Используйте этот бит:
Reset	22
AlarmHigh	23
AlarmLow	24
AlarmEn	25
OV	26
DN	27
LS	28
SA	29
FS	30
X	31

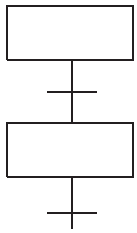
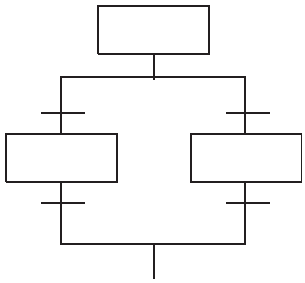
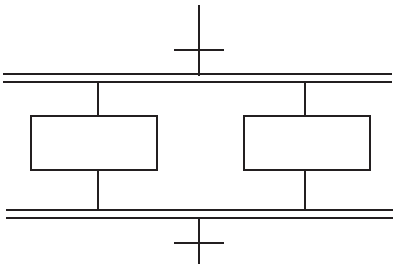
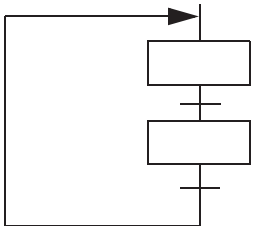
На следующей схема показана взаимосвязь битов X, FS, SA и LS.


step_name.X
step_name.FS
step_name.SA
step_name.LS
first scan
last scan

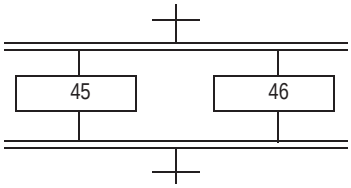
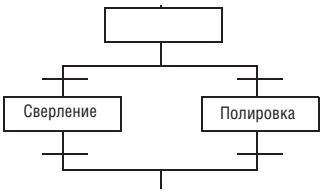
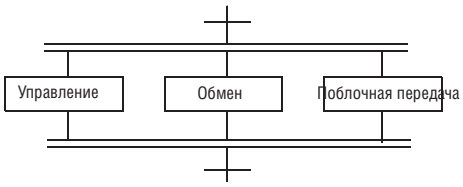
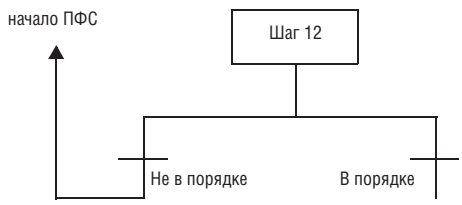
Организация шагов

Когда вы определите шаги своего процесса, организуйте их в виде последовательностей, одновременных ветвей, ветвей выбора или циклов.

Обзор

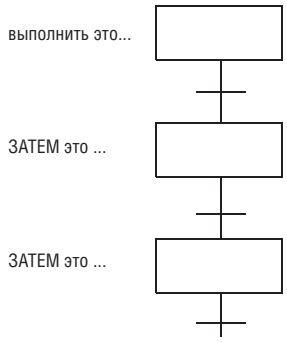
Для:	Используйте такую структуру:	С учетом следующего:
Последовательного выполнения одного или нескольких шагов: • Циклически выполняется один шаг. • Затем циклически выполняется следующий шаг.	**Последовательность** 	ПФС проверяет переход в конце шага: • Если «истина», ПФС переходит к следующему шагу. • Если «ложь», ПФС повторяет данный шаг.
• Выбора между альтернативными шагами или группами шагов в зависимости от условий логики • Выполнения какого-либо шага или шагов или пропуска какого-либо шага или шагов в зависимости от условий логики	**Ветвь выбора** 	• Допускается, чтобы путь не включал никаких шагов, а включал лишь переход. Это позволяет ПФС пропускать ветвь выбора. • По умолчанию ПФС проверяет переходы, с которых начинается каждый путь, слева направо. Она будет идти по первому истинному пути. • Если нет истинных переходов, ПФС повторяет предыдущий шаг. • Программное обеспечение RSLogix5000 позволяет изменять порядок, в котором ПФС осуществляет проверку переходов.
Одновременно выполнить 2 и более шагов. ПФС продолжится лишь после того, как все пути будут пройдены до конца.	**Одновременная ветвь** 	• Такая ветвь заканчивается одним переходом. • ПФС проверяет завершающий переход после как минимум однократного выполнения последнего шага в каждом пути. Если переход ложен, ПФС повторяет предыдущий шаг.
Вернуться к предыдущему шагу в цикле	**Связь, ведущая к предыдущему шагу** 	• Подведите связь к шагу или одновременной ветви, к которому вы хотите перейти. • Связь *не должна* входить в одновременную ветвь, выходить из нее или проходить внутри нее.

Далее приводится несколько примеров структур ПФС для различных случаев:

Пример ситуации:	**Пример решения:**
Станции 45 и 47 сборочной линии одновременно обрабатывают детали. По завершении обработки обеими станциями детали перемещаются на одну станцию ниже.	**Одновременная ветвь** 
В зависимости от кода станция выполняет сверление или полировку.	**Ветвь выбора** 
Для упрощения программы я хочу отделить обмен данными и поблочную передачу от другой управляющей логики. Все происходит одновременно.	**Одновременная ветвь** 
На участке термообработки температура увеличивается с заданной скоростью, затем достигнутая температура поддерживается в течение определенного времени, после чего происходит охлаждение с заданной скоростью.	**Последовательность** 
На станции 12 станок сверлит деталь, нарезает резьбу и сболчивает деталь. Шаги выполняются один за другим.	**Последовательность** 
Шаг 12 проверяет процесс на правильность смеси реагентов. Если все в порядке, то продолжить выполнение остальных шагов. Если нет, перейти в начало ПФС и очистить систему.	**Связь** 

Последовательность

Последовательность – это группа шагов, которые выполняются один за другим.



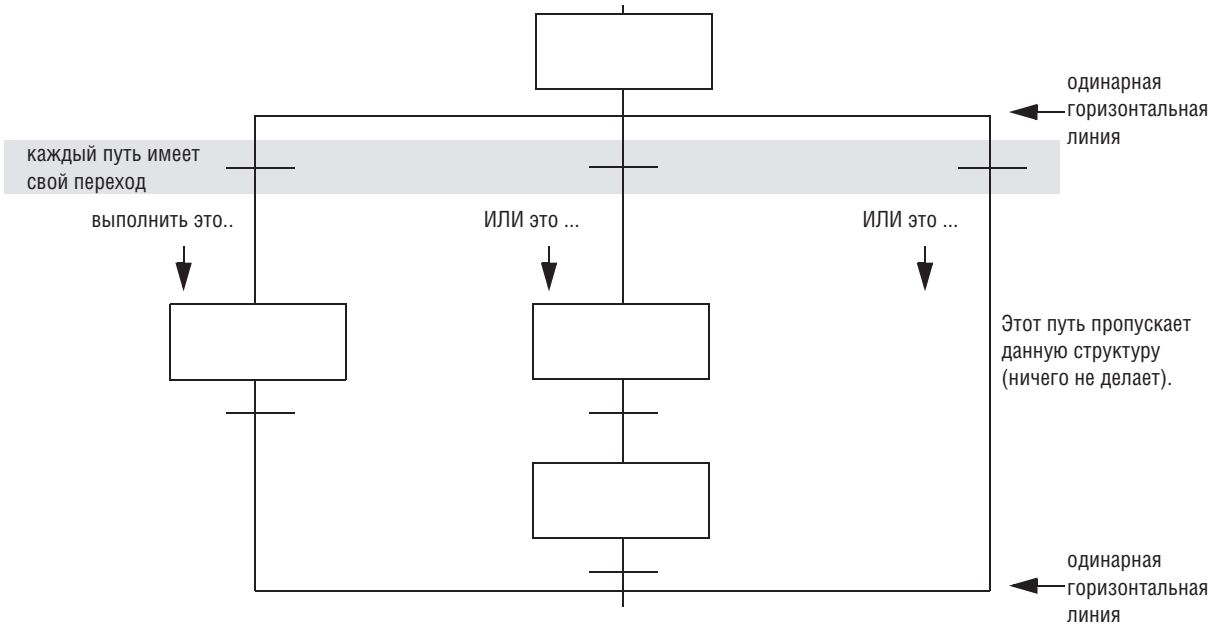
Подробная схема выполнения последовательности шагов приводится на рис. 5.5 на стр. 5-52.

Информацию о том, как обойти состояние перехода, вы найдете в разделе «Форсировка логических элементов» на стр. 14-1.

Ветвь выбора

Ветвь выбора представляет собой выбор между одним путем (шагом или группой шагов) и другим путем (т.е. структуру OR (ИЛИ)).

- Выполняется только один путь.
- По умолчанию ПФС проверяет переходы слева направо.
  - ПФС идет по первому истинному пути.
  - Программное обеспечение RSLogix5000 позволяет изменять порядок, в котором ПФС осуществляет проверку переходов. См. раздел «Программирование последовательной функциональной схемы» на стр. 6-1.



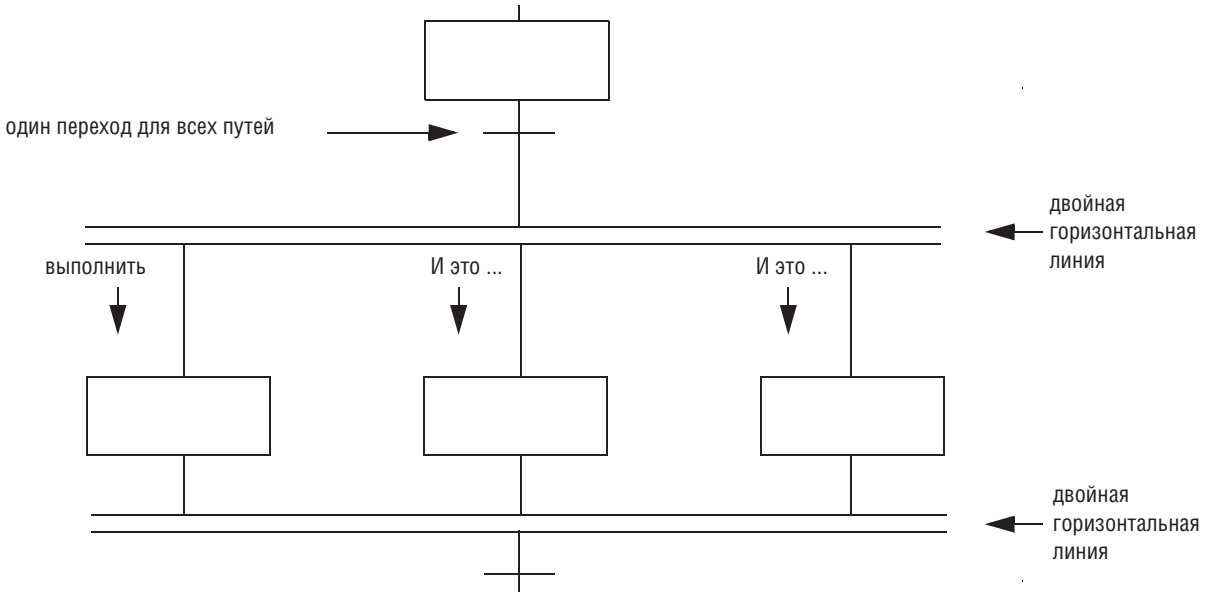
Подробная схема выполнения ветви выбора приводится на рис. 5.7 на стр. 5-54.

Информацию о том, как обойти состояние перехода, вы найдете в разделе «Форсировка логических элементов» на стр. 14-1.

Одновременная ветвь

Одновременная ветвь представляет собой пути (шаги или группы шагов), осуществляемые в одно и то же время (т.е. структуру AND (И)).

- Выполняются все пути.
- Выполнение ПФС будет продолжен только после завершения всех путей.
- ПФС проверяет переход после как минимум однократного выполнения последнего шага в каждом пути.



Подробная схема выполнения одновременной ветви приводится на рис. 5.6 на стр. 5-53.

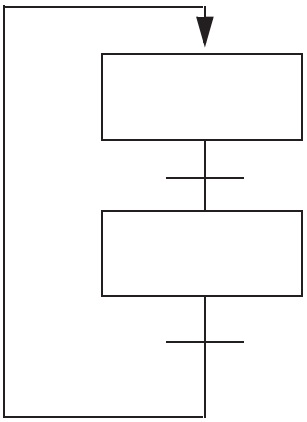
Информацию о том, как обойти ветвь и воспрепятствовать выполнению пути, вы найдете в разделе «Форсировка логических элементов» на стр. 14-1.

Связь, ведущая к предыдущему шагу

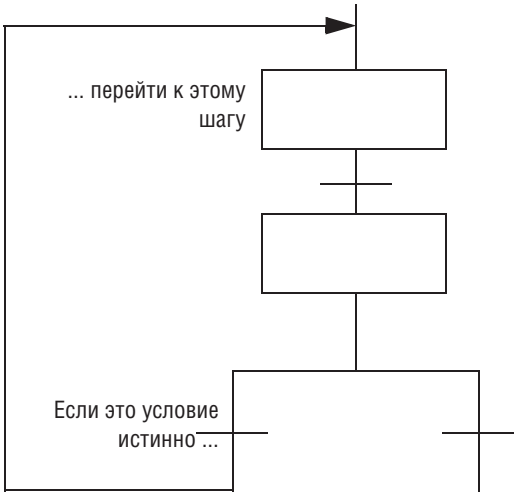
Помимо соединения шагов в виде последовательностей, одновременных ветвей и ветвей выбора вы можете соединить шаг с предыдущей точкой вашей ПФС. Это позволяет вам:

- возвращаться в начало цикла и повторять шаги
- возвращаться в начало ПФС и выполнять ее заново

Например:



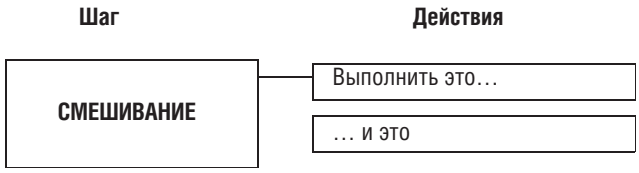
простой цикл, повторяющий выполнение всей ПФС



путь ветви выбора, осуществляющий возврат к предыдущему шагу

Добавление действий для каждого шага

Используйте **действия** для разделения шага на различные функции, выполняемые данным шагом, такие как подача команды двигателю, установка состояния клапана, или перевод группы устройств в определенный режим.



Как вы хотите использовать действие?

Существуют два типа действий:

Если вы хотите:	То:
выполнить структурированный текст непосредственно в ПФС	Используйте небулево действие
вызвать подпрограмму	
использовать опцию автоматического сброса для сброса данных по окончании шага	
просто установить бит и спрограммировать другую логику для контроля этого бита, чтобы определить, когда эта логика должна выполняться.	Используйте булево действие

Использование небулева действия

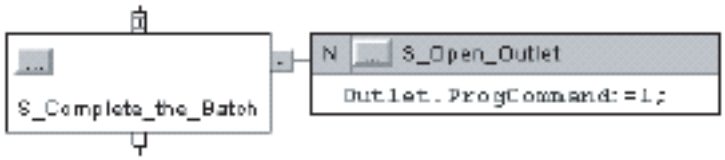
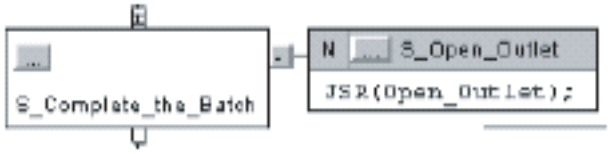
Небулево действие содержит логику для этого действия. Оно использует структурированный текст для выполнения присваиваний и инструкций или вызова подпрограммы.

При помощи небулева действия вы также можете выполнять **пост-сканирование** (автоматический сброс) присваиваний и инструкций перед выходом из шага:

- Во время пост-сканирования контроллер выполняет присваивания и инструкции таким образом, как если бы все условия были ложными.
- Контроллер выполняет пост-сканирование как встроенного структурированного текста, так и вызываемой действием подпрограммы.

Подробную информацию по автоматическому сбросу присваиваний и инструкций можно найти в разделе «Выключение устройства в конце шага» на стр. 5-32.

Вы можете спрограммировать небулево действие следующими способами:

Если вы хотите:	То:
• Выполнить свою логику без дополнительных процедур • использовать присваивания, конструкции и инструкции структурированного текста	Встройте структурированный текст. Например:  Когда шаг *S_Complete_the_Batch* активен, выполняется действие *S_Open_Outlet*. Это действие устанавливает тег *OutletProgCommand* равным 1, что открывает выходной клапан.
• повторно использовать логику в нескольких шагах • использовать другой язык для программирования действия, например, релейную логику • организовать вложение ПФС	Вызовите подпрограмму. Например:  Когда шаг *S_Complete_the_Batch* активен, выполняется действие *S_Open_Outlet*. Это действие вызывает процедуру *Open_Outlet*. **Процедура Open_Outlet**  При выполнении процедуры *Open_Outlet,* инструкция OTE устанавливает тег *OutletProgCommand* равным 1, что открывает выходной клапан.

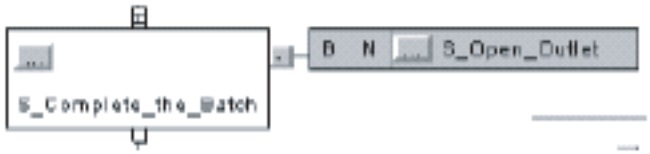
Небулево действие *нельзя* повторно использовать внутри одной и той же ПФС, за исключением его использования для сброса хранимого действия. На одну ПФС допускается лишь один экземпляр каждого небулева действия.

Использование булева действия

Булево действие не содержит никакой логики для этого действия. Оно просто устанавливает бит в своем теге (структура SFC_ACTION). Для выполнения такого действия другая логика должна контролировать соответствующий бит и выполняться при установленном бите.

При использовании булевых действий вы должны вручную сбрасывать присваивания и инструкции, связанные с соответствующим действием. Ввиду отсутствия связи между таким действием и выполняющей действие логикой, опция автоматического сброса не влияет на булевы действия.

Пример:



Когда шаг *S_Complete_the_Batch* активен, выполняется действие *S_Open_Outlet*. Когда это действие активно, его бит Q устанавливается.

Процедура релейной логики контролирует бит Q (*S_Open_Outlet.Q*). Когда бит Q установлен, инструкция JSR выполняется и открывает выходной клапан.

Вы можете многократно повторно использовать булево действие внутри одной и той же ПФС.

Структура SFC_ACTION

Каждой действие (небулево и булево) использует тег, содержащий информацию об этом действии. Вы можете обратиться к этой информации посредством диалогового окна *Action Properties* (Свойства действия) или закладки *Monitor Tags* (Контроль тегов) в окне *Tags* (Теги):

Если вы хотите:	То проверьте или установите этот член:	Тип данных:	Описание:	
определить, когда действие активно	Q	BOOL	Состояние бита Q зависит от того, является ли данное действие булевым или небулевым:	
			Если действие является:	**То бит Q:**
			булевым	установлен (1) все время, пока действие активно, включая его последнее сканирование
			небулевым	установлен (1), пока действие активно, но сброшен (0) при последнем сканировании данного действия
			Если вы хотите использовать бит для определения, когда действие активно, используйте бит Q.	
	A	BOOL	Бит A активен все время, пока действие активно.	
определить, сколько времени действие было активно (в миллисекундах)	T	DINT	Когда действие становится активным, значение таймера (Timer (T)) сбрасывается, после чего вновь начинается отсчет времени в миллисекундах. Отсчет времени продолжается до того момента, когда действие станет неактивным, независимо от значения уставки (Preset (PRE)).	
использовать один из следующих определителей, основанных на времени: L, SL, D, DS, SD	PRE	DINT	Введите временной предел или задержу а член Preset (PRE). Действие будет начинаться или прекращаться при достижении значением Timer (T) значения уставки (Preset). Вы также можете ввести численное выражение, вычисляющее время в процессе выполнения.	
определить, сколько раз действие становилось активным	Count	DINT	Это *не* является подсчетом числа сканирований данного действия. • Счетчик увеличивается на единицу каждый раз, когда действие становится активным. • Он вновь увеличивается на единицу только после того, как действие становится неактивным, а затем вновь становится активным. • Счетчик сбрасывается только в том случае, если вы сконфигурируете ПФС на перезапуск в начальном шаге. При такой настройке он будет сбрасываться при переходе контроллера из программного режима в режим выполнения.	
использовать один тег для различных битов состоянияданного действия	Status	DINT	**Для этого члена:**	**Используйте этот бит:**
			Q	30
			A	31

Описание каждого действия в псевдокоде

Чтобы организовать логику для какого-либо действия, в первую очередь опишите действие в псевдокоде. Если вы не знакомы с псевдокодом, придерживайтесь следующих указаний:

- Используйте ряд коротких предложений, точно описывающих, что должно происходить.

- Используйте такие термины или обозначения как if (если), then (то), otherwise (иначе), until (до тех пор пока), and (и), or (или), =, >, <.
- Расположите предложения в порядке выполнения.
- При необходимости укажите, какие условия должны проверяться сначала (when 1st (когда вначале)), и какое действие должно за этим следовать (what 2nd (что потом)).

Введите псевдокод в тело соответствующего действия. После этого вы можете:

- Доработать псевдокод таким образом, чтобы он выполнялся как структурированный текст.
- Использовать псевдокод для разработки вашей логики и оставить его в качестве комментариев. Поскольку все комментарии структурированного текста загружаются в контроллер, ваш псевдокод всегда будет доступен в качестве документального описания действия.

Для преобразования псевдокода в комментарии структурированного текста добавьте следующие символы комментария:

Для комментария:	**Используйте один из следующих форматов:**
на одной строке	// комментарий
занимающего более одной строки	(*начало комментария ... конец комментария*) /*начало комментария ... конец комментария*/

Выбор определителя для действия

Каждое действие (небулево и булево) использует **определитель** для задания момента его начала и окончания.

По умолчанию используется определитель *Non-Stored* (Без сохранения). Действие запускается при активизации шага и останавливается, когда шаг становится неактивным.

Чтобы изменить момент начала и окончания действия, задайте другой определитель:

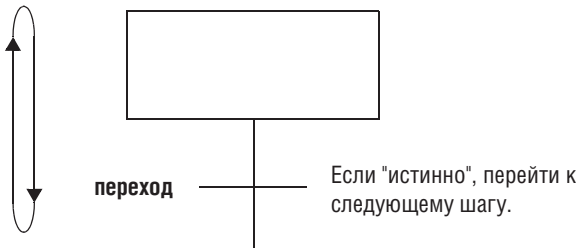
Таблица 5.1 Выбор определителя для действия

Если вы хотите, чтобы действие:	И:	То задайте этот определитель:	Который расшифровывается как:
запускалось при активизации шага	останавливалось, когда шаг становится неактивным	N	Non-Stored (Без сохранения)
	выполнялось только один раз	P1	Pulse (Rising Edge) (Импульс (передний фронт))
	останавливалось до того, как шаг станет неактивным или в тот момент, когда шаг становится неактивным	L	Time Limited (Временное ограничение)
	оставалось активным до тех пор, пока действие *Reset* не выключит данное действие	S	Stored (С сохранением)
	оставалось активным до тех пор, пока действие *Reset* не выключит данное действие или пока не истечет заданное время, даже если шаг станет неактивным	SL	Stored and Time Limited (С сохранением и ограничением по времени)
запускалось через определенное время после активизации шага *и* в то время, пока шаг все еще активен	останавливалось, когда шаг становится неактивным	D	Time Delayed (С временной задержкой)
	оставалось активным до тех пор, пока действие *Reset* не выключит данное действие	DS	Delayed and Stored (С временной задержкой и сохранением)
запускалось через определенное время после активизации шага, даже если шаг уже стал неактивным	оставалось активным до тех пор, пока действие *Reset* не выключит данное действие	SD	Stored and Time Delayed (С сохранением и временной задержкой)
однократно выполнялось при активизации шага	однократно выполнялось, когда шаг становится неактивным	P	Pulse (Импульс)
запускалось, когда шаг становится неактивным	выполнялось только один раз	P0	Pulse (Falling Edge) (Импульс(задний фронт))
отключало (сбрасывало) хранимое действие: • S Stored • SL Stored and Time Limited • DS Delayed and Stored • SD Stored and Time Delayed		R	Reset (Сброс)

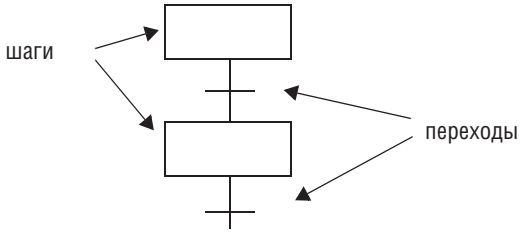
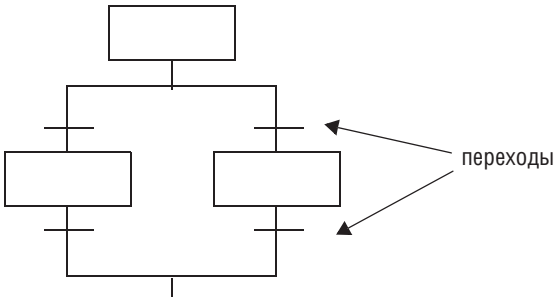
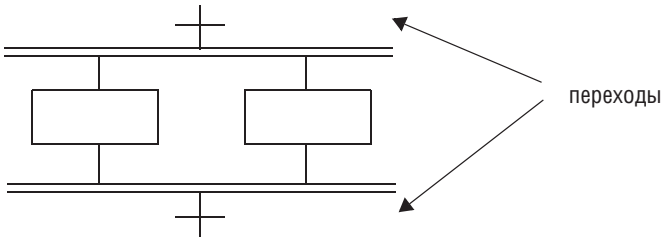
Задание условий перехода

Переход – это физические условия, которые должны иметься или измениться, чтобы произошел переход к следующему шагу.

Переход указывает ПФС, когда переходить к следующему шагу.



Переходы могут находиться в следующих местах:

Для этой структуры:	Убедитесь в том, что:
последовательность	Переход имеется между всеми шагами. 
ветвь выбора	Переходы находятся внутри участка, ограниченного горизонтальными линиями. переходы 
одновременная ветвь	Переходы находятся снаружи горизонтальных линий. переходы 

Далее приводятся примеры переходов:

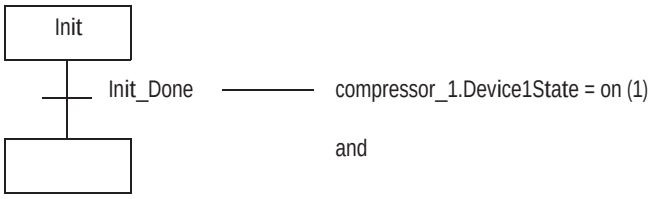
ПРИМЕР

Вы хотите:

а. Включить 2 компрессора. При включенном компрессоре должен быть установлен бит Device1State.

б. Когда оба компрессора будут включены, перейти к следующему шагу.

Решение:



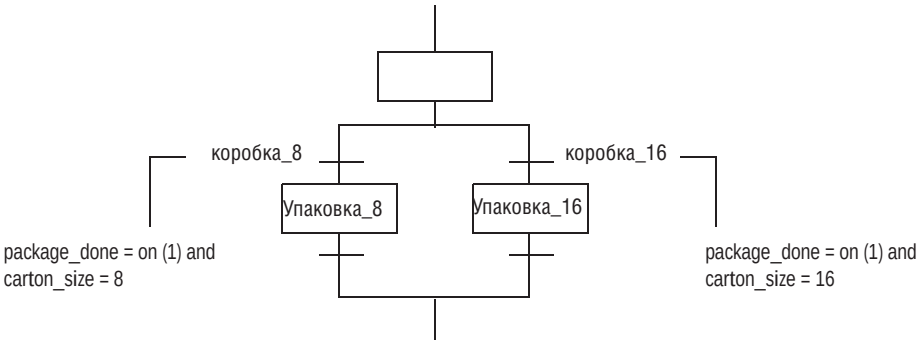
ПРИМЕР

Вы хотите:

а. Упаковать продукт. Когда продукт находится в упаковке, устанавливается бит *package_done*.

б. Упаковывать продукт по 8 штук в коробке или по 16 штук в коробке.

Решение:



Информация по переопределению состояния перехода содержится в разделе «Форсировка логических элементов» на стр. 14-1.

Тег перехода

Каждый переход использует тег типа BOOL для отображения истинного или ложного состояния перехода.

Если переход является:	То тег принимает значение:	И:
истинным	1	ПФС переходит к следующему шагу.
ложным	0	ПФС продолжает выполнение текущего шага.

Как вы хотите запрограммировать переход?

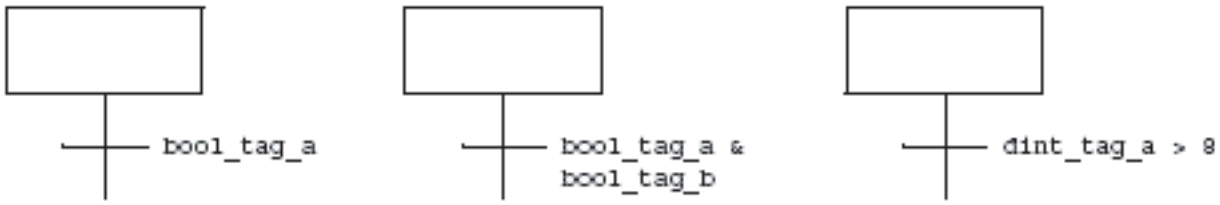
Вы можете запрограммировать переход следующими способами:

Если вы хотите:	То:
ввести условия в структурированный текст в виде выражения	Используйте выражение BOOL
ввести условия в другую процедуру в виде инструкций	Вызовите подпрограмму
использовать одну и ту же логику для нескольких переходов	

Использование выражения BOOL

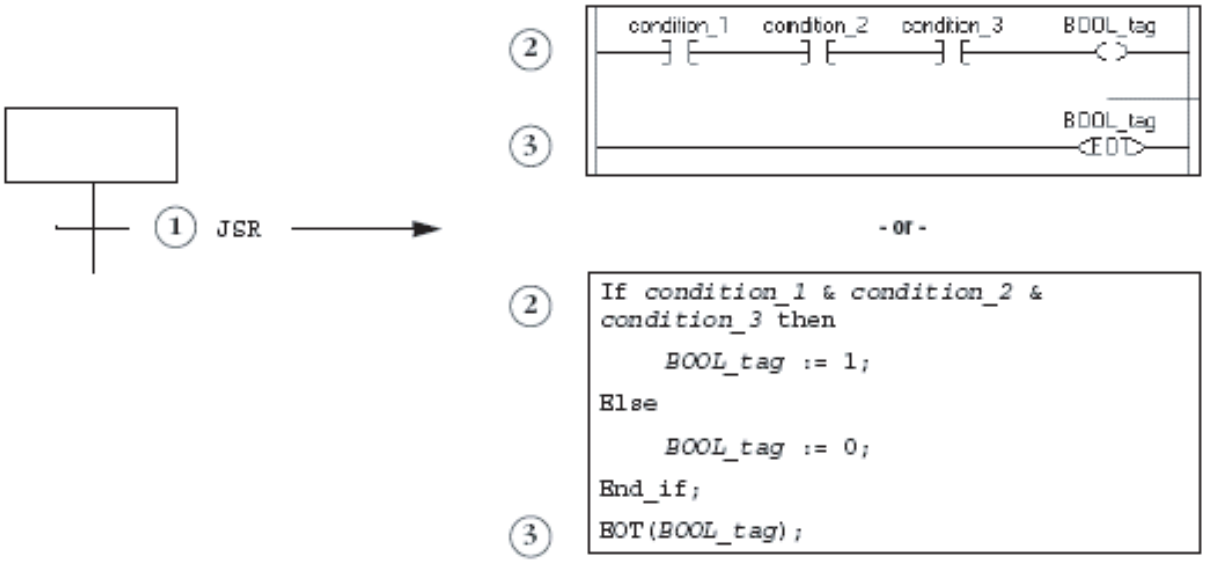
Самым простым способом программирования перехода является ввод условий в структурированный текст в виде **выражения BOOL.** Выражение BOOL использует булевы теги, операторы отношения и логические операторы для сравнения значений или проверки условий на истинность или ложность. Например, `tag1>65`.

Ниже приводятся несколько примеров выражений BOOL.



Вызов подпрограммы

Чтобы использовать подпрограмму для управления переходом, включите в нее инструкцию End OF Transition (EOT) (Конец перехода). Эта инструкция возвращает переходу состояние его условий как показано ниже.



1.. Вызовите подпрограмму.

2. Проверьте необходимые условия. Когда эти условия истинны, устанавливается тег типа BOOL

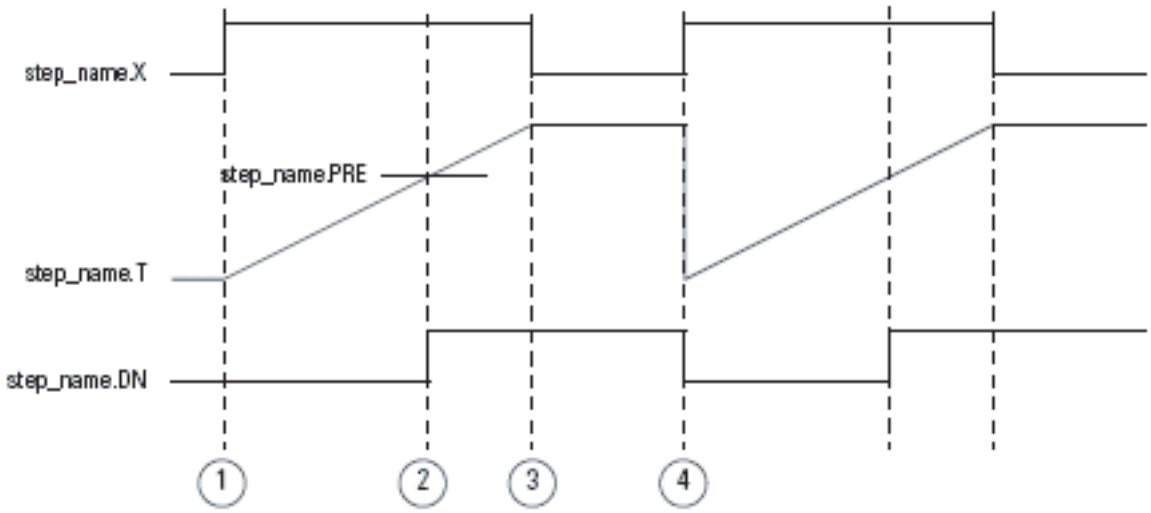
3. Используйте инструкцию EOT для того, чтобы установить состояние перехода равным значению тега типа BOOL. Когда этот тег установлен (истинен), переход является истинным.

Переход по истечении заданного времени

Каждый шаг в ПФС включает миллисекундный таймер, работающий всегда, когда шаг активен. Используйте этот таймер для:

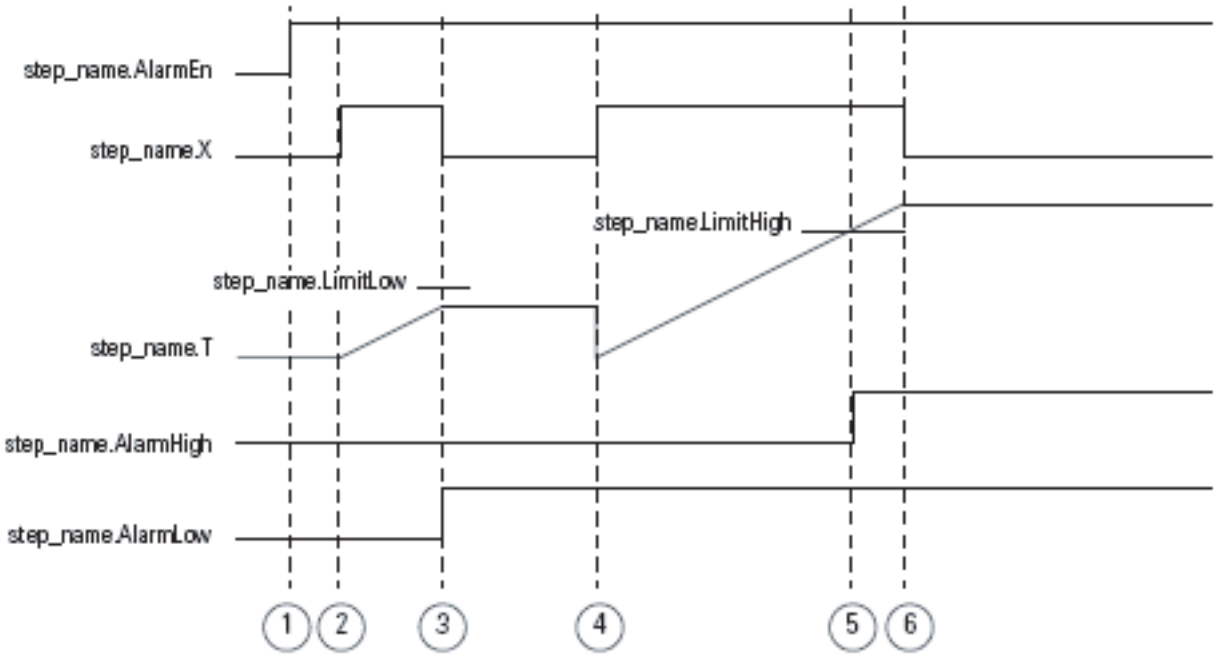
- сигнализации того, что данный шаг проработал требуемое время и ПФС надлежит перейти к следующему шагу
- сигнализации того, что шаг выполняется слишком долго и ПФС надлежит перейти к шагу ошибки

Рисунок 5.3 На следующей схеме показано действие таймера и связанные с ним биты шага:



Описание:
1. Шаг становится активным. Бит X устанавливается. Значение таймера Timer (T) начинает увеличиваться.
2. Таймер достигает значения уставки Preset (PRE) для данного шага. Бит DN устанавливается. Значение таймера продолжает увеличиваться.
3. Шаг становится неактивным. Бит X сбрасывается. Значение таймера не изменяется. Бит DN остается установленным.
4. Шаг становится активным. Бит X устанавливается. Значение таймера обнуляется, затем начинает увеличиваться. Бит DN сбрасывается.

Рисунок 5.4 На следующей схеме показано, как работает сигнализация нижнего и верхнего предела для шага



:

Описание:
1. Бит AlarmEn установлен. Установите этот бит, чтобы использовать сигнализацию нижнего и верхнего предела. Это можно сделать посредством диалогового окна свойств (Properties) или при помощи тега для данного шага.
2. Шаг становится активным. Бит X устанавливается. Значение таймера Timer (T) начинает увеличиваться.
3. Шаг становится неактивным. Бит X сбрасывается. Значение таймера не изменяется. Поскольку значение Timer меньше значения LimitLow (нижний предел), устанавливается бит AlarmLow (сигнализация нижнего предела).

Описание:
4. Шаг становится активным. Бит X устанавливается. Значение таймера обнуляется, затем начинает увеличиваться. Бит AlarmLow остается установленным. (Вам необходимо сбросить его вручную).
5. Таймер достигает значения LimitHigh (Верхний предел) для данного шага. Устанавливается бит AlarmHigh (Сигнализация верхнего предела). Значение таймера продолжает увеличиваться.
6. Шаг становится неактивным. Бит X сбрасывается. Значение таймера не изменяется. Бит AlarmHigh остается включенным. (Вам необходимо сбросить его вручную.)

Ниже приводится пример использования уставки времени для шага.

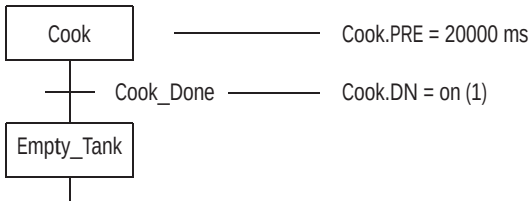
ПРИМЕР

В функциональной спецификации указывается:

а. Готовить ингредиенты в баке в течение 20 секунд.

б. Опорожнить бак.

Решение:



Далее приводится пример использования сигнализации верхнего предела для шага.

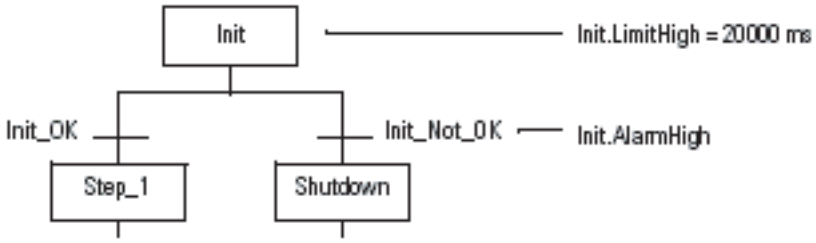
ПРИМЕР

В функциональной спецификации указывается:

а. Привести в исходное состояние 8 устройств.

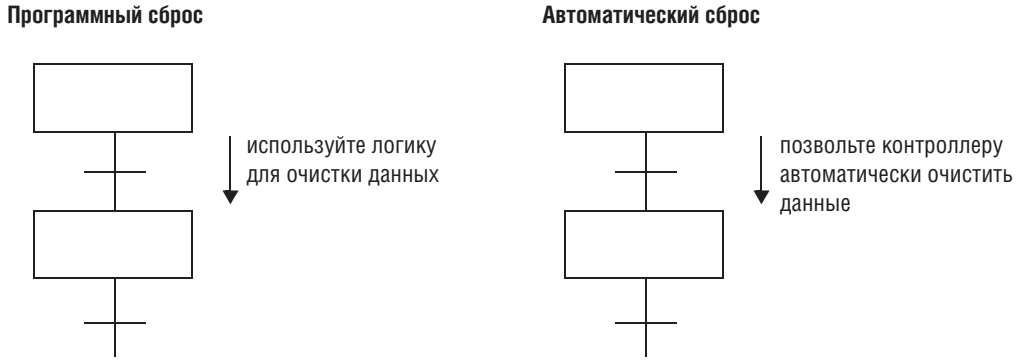
б. Если все 8 устройств не вернутся в исходное состояние в течение 20 секунд, то остановить систему.

Решение:



Выключение устройства в конце шага

При выходе ПФС из шага вы можете воспользоваться несколькими способами выключения устройств, включенных данным шагом.



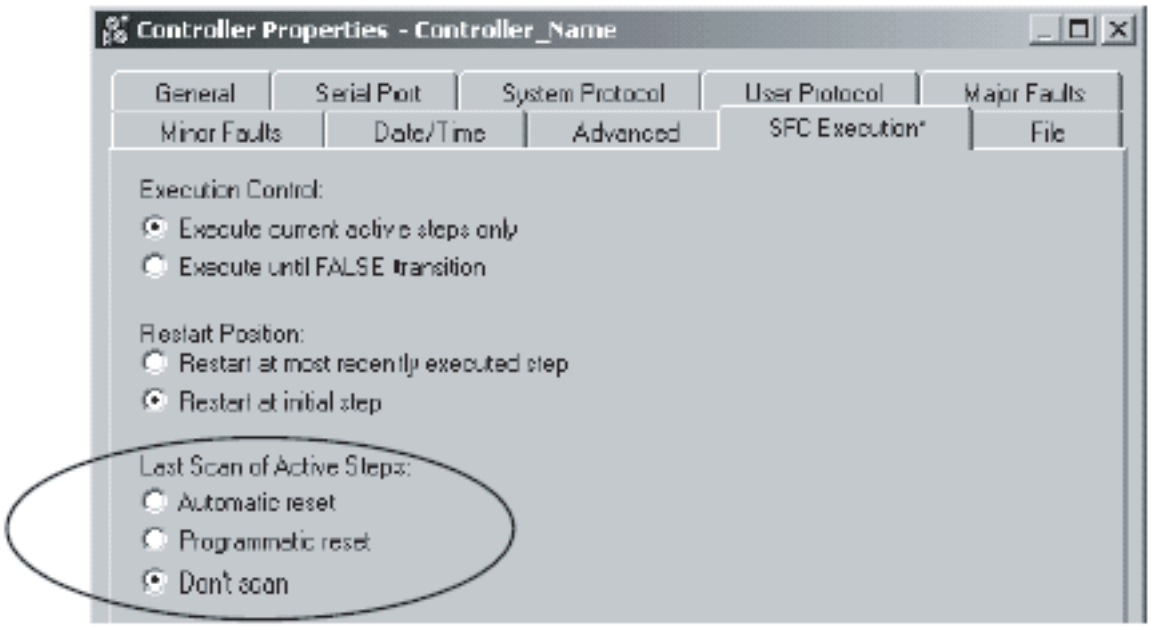
Каждый из этих вариантов требует от вас:

1. Выбрать опцию последнего сканирования.
2. Исходя из опции последнего сканирования, разработать свою логику так, чтобы последнее сканирование возвращало данные к нужным значениям.

Выбор опции последнего сканирования

Для последнего сканирования каждого шага вы можете воспользоваться следующими опциями. Выбранная вами опция применяется для всех шагов ПФС данного контроллера.

Если вы хотите:	И во время последнего сканирования шага:	То:	См. стр.:
управлять тем, какие данные должны быть очищены	Выполнять *только* действия P и P0 и использовать их для очистки соответствующих данных.	Используйте опцию Don't Scan (Не сканировать)	5-34
	Выполнять *все* действия и использовать один из следующих способов очистки соответствующих данных: • биты состояния данного шага или действия для обусловливания логики • действия P и P0	Используйте опцию Programmatic Reset (Программный сброс)	5-35
позволить контроллеру выполнять очистку данных	→	Используйте опцию Automatic Reset (Автоматический сброс)	5-38


Controller Properties - Controller_Name
General
Serial Port
System Protocol
User Protocol
Major Faults
Minor Faults
Date/Time
Advanced
SFC Execution*
File
Execution Control:
Execute current active steps only
Execute until FALSE transition
Restart Position:
Restart at most recently executed step
Restart at initial step
Last Scan of Active Steps:
Automatic reset
Programmatic reset
Don't scan

В следующей таблице сравниваются различные опции выполнения последнего сканирования шага:

Характеристика:	При использовании этой опции для последнего сканирования шага происходит следующее:		
	Don't scan (Не сканировать)	**Programmatic reset (Программный сброс)**	**Automatic reset (Автоматический сброс)**
выполняемые действия	Выполняются только действия P и P0. Они выполняются в соответствии с их логикой.	Все действия выполняются в соответствии с их логикой.	• Действия P и P0 выполняются в соответствии с их логикой. • Все остальные действия выполняются в режиме пост-сканирования. • Во время следующего сканирования данной процедуры действия P и P0 выполняются в режиме пост-сканирования.
сохранение значений	Все данные сохраняют свои текущие значения.	Все данные сохраняют свои текущие значения.	• Данные возвращаются к своим значениям для пост-сканирования. • Теги в левой части присваиваний [:=] обнуляются.
способ очистки данных	Используйте действия P и P0.	Используйте один из следующих способов: • биты состояния данного шага или действия для обусловливания логики • действия P и P0	Используйте один из следующих способов: • присваивание [:=] (не сохраняющее присваивание) • инструкции, очищающие свои данные во время пост-сканирования
сброс вложенной ПФС	Вложенная ПФС остается на своем текущем шаге.	Вложенная ПФС остается на своем текущем шаге.	Если вы выберите опцию *Restart at initial step* (Перезапуск с начального шага) для свойства *Restart Position* (Позиция перезапуска), то: • Вложенная ПФС вернется к своему начальному шагу. • Бит X стопового элемента во вложенной ПФС обнулится.

Использование опции Don't Scan (Не сканировать)

По умолчанию для последнего сканирования шага используется опция *Don't scan* (Не сканировать). При использовании этой опции все данные сохраняют свои текущие значения при выходе ПФС из шага. Поэтому вы должны использовать дополнительные присваивания или инструкции для очистки всех данных, которые вы хотите отключить в конце шага.

Чтобы выключить устройство в конце шага:

1. Убедитесь в том, что свойство *Last Scan of Active Steps* (Последнее сканирование активных шагов)установлено на опцию *Don't scan* (по умолчанию).

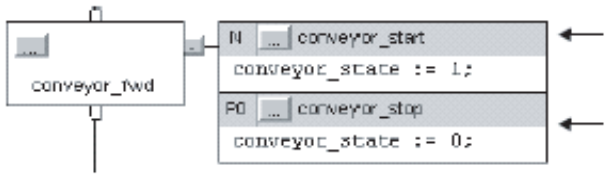
2.. Используйте действие *P0 Pulse (Falling Edge)* (Задний фронт импульса) для очистки соответствующих данных. Убедитесь в том, что действие или действия P0 являются последними в последовательности действий для данного шага.

При использовании опции *Don't scan* для последнего сканирования шага выполняются только действия P и P0. Присваивания и инструкции этих действий выполняются в соответствии с условиями их логики.

- Контроллер *не* выполняет **пост-сканирование** присваиваний и инструкций.
- Когда ПФС выходит из данного шага, все данные сохраняют свои текущие значения.

В следующем примере используется действие для включения конвейера в начале шага. Другое действие выключает конвейер в конце этого шага.

ПРИМЕР Использование опции Don't Scan



Это действие включает конвейер. Когда *conveyor_state* устанавливается, конвейер включается.

Перед выходом ПФС из данного шага действие P0 выключает конвейер. Во время последнего сканирования шага *conveyor_state* сбрасывается. Это приводит к выключению конвейера.

Использование опции Programmatic Reset (Программный сброс)

Способ программного выключения (сброса) устройств в конце шага заключается в выполнении всех действий при последнем сканировании шага. Это позволяет вам выполнить вашу обычную логику и в то же время выключить (сбросить) устройства в конце шага.

1. В свойстве *Last Scan of Active Steps* (Последнее сканирование активных шагов) выберите опцию *Programmatic reset* (Программный сброс).

2. Очистите соответствующие данные одним из следующих способов:

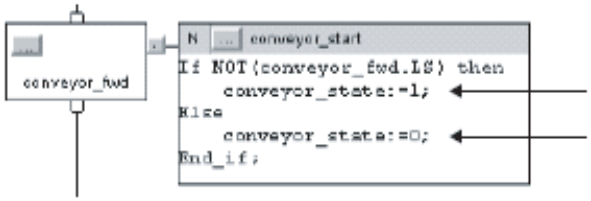
- К вашей обычной логике добавьте логику, очищающую соответствующие данные. Используйте бит LS данного шага или бит Q соответствующего действия для обусловливания выполнения логики.
- Используйте действие *P0 Pulse (Falling Edge)* (Задний фронт импульса)для очистки соответствующих данных. Убедитесь в том, что действие или действия P0 являются последними в последовательности действий для данного шага.

При использовании опции *Programmatic reset* для последнего сканирования шага выполняются все присваивания и инструкции в соответствии с условиями логики.

- Контроллер *не* выполняет **пост-сканирование** присваиваний и инструкций.
- Когда ПФС выходит из данного шага, все данные сохраняют свои текущие значения.

В следующем примере используется одно действие для включения и выключения конвейера. Бит LS шага обусловливает выполнение логики. См. раздел «Структура SFC-STEP» на странице 5-8.

ПРИМЕР Использование опции Programmatic Reset и бита LS

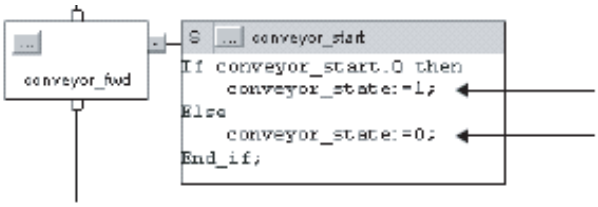


Когда шаг находится не на последнем сканировании (*conveyor_fwd.LS* = 0), этот оператор устанавливает *conveyor_state*. Когда *conveyor_state* устанавливается, конвейер включается.

При последнем сканировании данного шага (*conveyor_fwd.LS* = 1) этот оператор сбрасывает *conveyor_state*. Когда *conveyor_state* сбрасывается, конвейер выключается.

Для действия, использующего один из сохраняемых определителей, используйте бит Q этого действия для обусловливания вашей логики. См. раздел «Структура SFC_ACTION» на странице 5-20.

ПРИМЕР Использование опции Programmatic Reset и бита Q

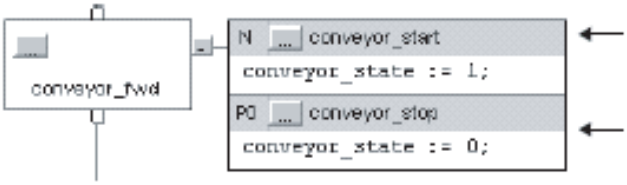


Когда шаг находится не на последнем сканировании (*conveyor_start.Q* = 1), этот оператор устанавливает *conveyor_state*. Когда *conveyor_state* устанавливается, конвейер включается.

При последнем сканировании данного шага (*conveyor_start.Q* = 0) этот оператор сбрасывает *conveyor_state*. Когда *conveyor_state* сбрасывается, конвейер выключается.

Также для очистки данных вы можете использовать действие *P0 Pulse (Falling Edge)* (Задний фронт импульса). В следующем примере это действие используется для включения конвейера в начале шага. Другое действие выключает конвейер в конце данного шага.

ПРИМЕР Использование опции Programmatic Reset и действия P0



Это действие включает конвейер. Когда *conveyor_state* устанавливается, конвейер включается.

Перед выходом ПФС из данного шага действие P0 выключает конвейер. Во время последнего сканирования шага *conveyor_state* сбрасывается. Это приводит к выключению конвейера.

Использование опции Automatic Reset (Автоматический сброс)

Для автоматического выключения (сброса) устройств в конце шага:

1. В свойстве *Last Scan of Active Steps* (Последнее сканирование активных шагов) выберите опцию *Automatic reset* (Автоматический сброс).
2. Для выключения устройства в конце шага управляйте его состоянием с помощью присваиваний или инструкций, таких как:
   - присваивание [:=] (не сохраняющее присваивание)
   - инструкция Output Energize (OTE) в подпрограмме

При использовании опции *Automatic reset* во время последнего сканирования происходит следующее:

- выполняются действия P и P0 в соответствии с условиями их логики
- обнуляются теги в левой части присваиваний [:=]
- выполняется **пост-сканирование** встроенного структурированного текста
- выполняется пост-сканирование всех подпрограмм, вызываемых действием при помощи инструкции Jump to Subroutine (JSR)
- сбрасываются все вложенные ПФС (ПФС, вызываемые действием как подпрограммы)

ВАЖНО Пост-сканирование действия фактически происходит тогда, когда действие переходит из активного состояния в неактивное. В зависимости от определителя действия пост-сканирование может происходить перед последним сканированием соответствующего шага или после него.

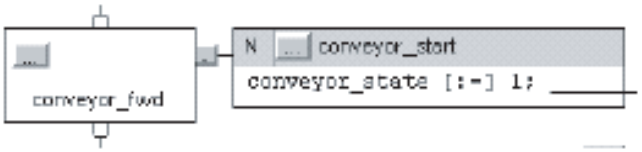
Как правило, пост-сканирование выполняет инструкции так, как если бы все условия были ложными. Например, инструкция Output Energize (OTE) в процессе пост-сканирования очищает свои данные.

Некоторые инструкции *не* следуют этому общему правилу при пост-сканировании. За описанием выполнения конкретной инструкции при пост-сканировании обращайтесь к следующим руководствам:

- Справочное руководство по общим инструкциям для контроллеров Logix5000 (Logix5000 Controllers General Instructions Reference Manual), публикация 1756-RM003
- Справочное руководство по инструкциям управления процессом и приводами для контроллеров Logix5000 (Logix5000 Controllers Process and Drives Instructions Reference Manual), публикация 1756-RM006
- Справочное руководство по набору инструкций управления перемещением для контроллеров Logix5000 (Logix5000 Controllers Motion Instruction Set Reference Manual), публикация 1756-RM007

Ниже приводится пример использования не сохраняющего присваивания для управления конвейером. Оно включает конвейер в начале шага и автоматически выключает его по окончании выполнения шага.

ПРИМЕР Автоматическая очистка данных

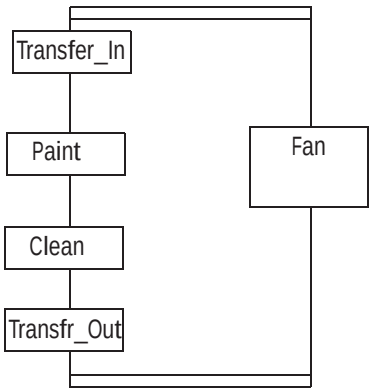
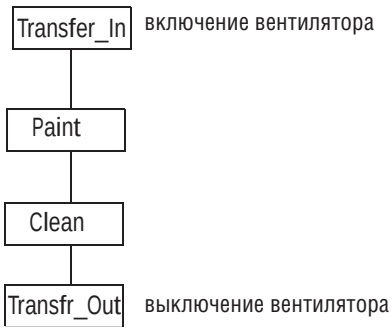
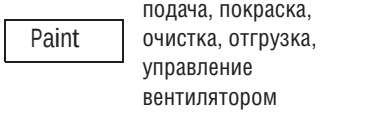


Это действие включает конвейер. Когда *conveyor_state* устанавливается, конвейер включается.

Поддержание чего-либо во включенном состоянии от шага к шагу

Как вы хотите управлять устройством?

Для обеспечения бесперебойного управления каким-либо устройством в течение нескольких периодов времени или фаз (шагов) воспользуйтесь одним из следующих способов:

Способ:	Пример:
Использование одновременной ветви Сделайте отдельный шаг, управляющий устройством.	
Сохранение и сброс действия Отметьте шаг, включающий устройство, и шаг, выключающий это устройство. Затем задайте пару действий - хранимое (Stored) и сброс (Reset) - для управления этим устройством.	
Использование одного укрупненного шага Используйте один укрупненный шаг, содержащий все действия, происходящие при включенном устройстве.	

Использование одновременной ветви

Простым способом управления устройством или устройствами в течение одного или нескольких шагов является создание отдельного шага для этих устройств и использование одновременной ветви для выполнения этого шага в течение всего остального процесса.

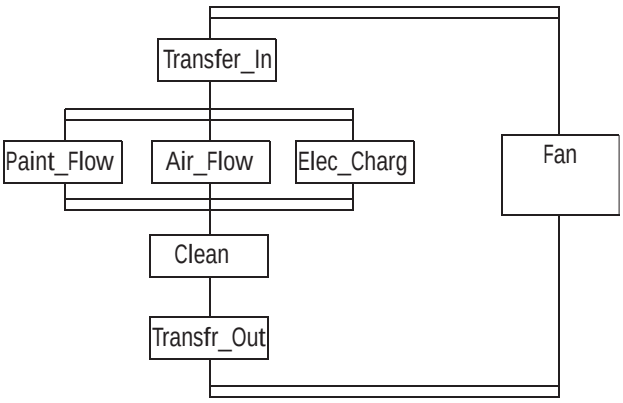
Далее приводится пример использования такого способа.

ПРИМЕР

Операция покраски включает следующее:

1. Подачу изделия в цех покраски.
2. Покраску изделия при помощи трех пистолетов-распылителей.
3. Очистку пистолетов.
4. Транспортировку изделия в сушильные печи.

Решение:

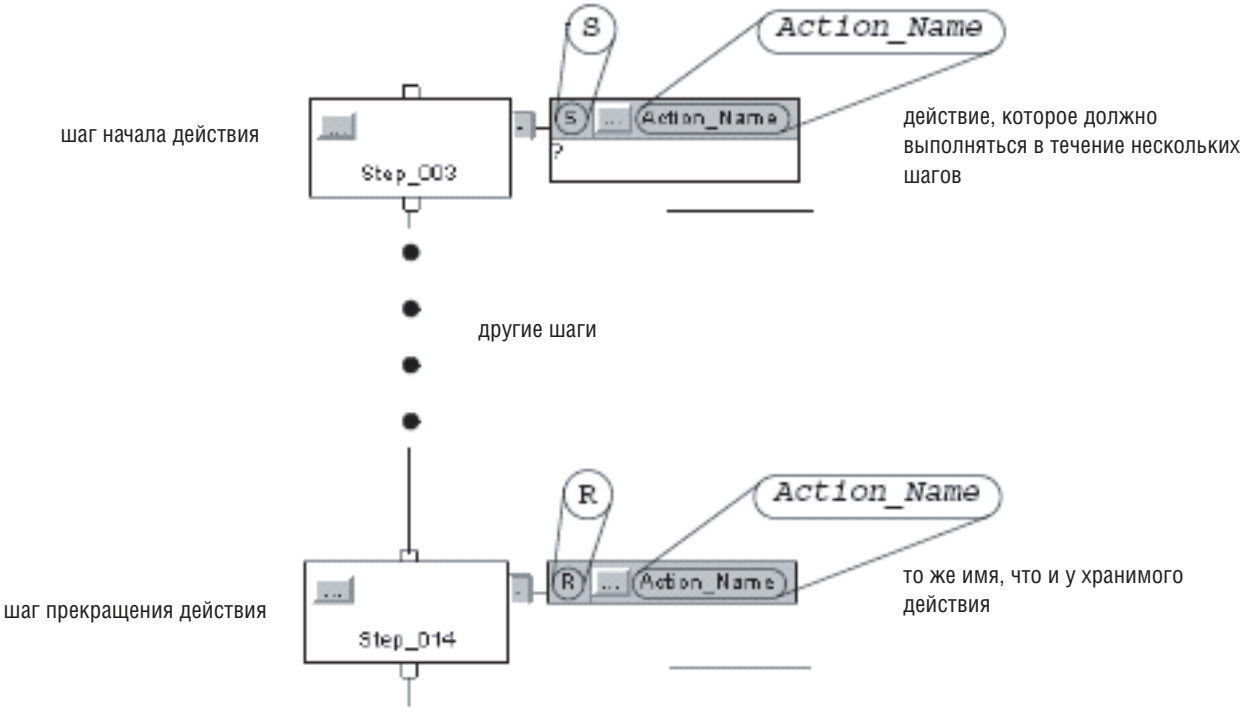


Сохранение и сброс действия

Обычно действие выключается (прекращает выполняться) при переходе ПФС к следующему шагу. Для бесперебойного поддержания устройства во включенном состоянии от шага к шагу сохраните действие, управляющее этим устройством:

1. В шаге, включающем устройство, назначьте управляющему устройством действию определитель с сохранением. Перечень определителей с сохранением содержится в Таблице 5.1 на странице 5-23.
2. В шаге, выключающем это устройство, используйте действие *Reset* (Сброс).

На следующем рисунке показано использование хранимого действия.

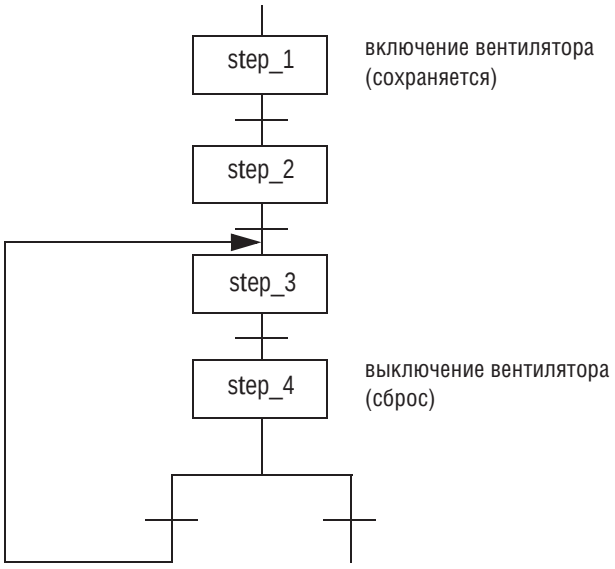


Когда ПФС выходит из шага, где хранится данное действие, программное обеспечение RSLogix 5000 продолжает показывать хранимое действие как активное. (По умолчанию это действие заключено в зеленую рамку.) Это информирует вас о том, что ПФС выполняет логику данного действия.

При использовании хранимого действия придерживайтесь следующих указаний:

- Действие *Reset* выключает лишь хранимое действие. Оно *не* выключает автоматически устройства, относящиеся к данному действию. Для выключения устройства за действием *Reset* должно следовать другой действие, выключающее это устройство. Другим способом является использование опции *Automatic reset* (Автоматический сброс), описанной на странице 5-38.
- Перед тем, как ПФС дойдет до стопового элемента, сбросьте все хранимые действия, которые *не* должны выполняться при стопе ПФС. Всякое активное сохраненное действие остается активным даже при достижении ПФС стопа.
- С осторожностью используйте переход между шагом, где хранится действие, и шагом, сбрасывающему это действие. После сброса действия оно запускается только при выполнении того шага, где хранится это действие.

  В следующем примере для шагов 1-4 требуется включенный вентилятор. В конце шага s*tep_4* вентилятор сбрасывается (выключается). Когда ПФС возвращается к шагу s*tep_3*, вентилятор остается выключенным.



Для включения вентилятора ПФС должна вернуться к шагу *step_1*.

Использование одного укрупненного шага

Если вы используете один укрупненный шаг для нескольких функций, то используйте дополнительную логику для задания порядка выполнения этих функций. Одним из вариантов является вложение ПФС внутри такого укрупненного шага.

В следующем примере шаг включает вентилятор, а затем вызывает другую ПФС. Эта вложенная ПФС определяет порядок выполнения остальных функций шага. Вентилятор остается включенным в течение всех шагов вложенной ПФС.

ПРИМЕР Использование укрупненного шага

Это действие включает вентилятор:

- *fan.ProgProgReq* позволяет ПФС управлять состоянием вентилятора.
- *fan.ProgCommand* включает вентилятор.

Это действие вызывает другую ПФС, которая определяет порядок выполнения остальных функций этого шага.

За дополнительной информацией по организации вложенных ПФС обращайтесь к разделу «Вложение ПФС» на странице 5-49.

Окончание ПФС

После выполнения ПФС своего последнего шага она *не* перезапускается автоматически с первого шага. Вы должны указать ПФС, что нужно делать по окончании последнего шага.

Что вы хотите делать в конце ПФС?

Для:	Сделайте следующее:
автоматического возврата к какому-либо из предыдущих шагов	Проведите связь от последнего перехода к верхней части шага, к которому вы хотите перейти. См. раздел «Связь, ведущая к предыдущему шагу» на стр. 5-17.
останова и ожидания команды на перезапуск	Используйте стоповый элемент. См. раздел «Использование стопового элемента» на стр. 5-45.

Использование стопового элемента

Стоповый элемент позволяет остановить выполнение всей ПФС или пути одновременной ветви и подождать перезапуска. Когда ПФС доходит до стопового элемента, происходит следующее:

- Бит X стопового элемента устанавливается. Это сигнализирует о том, что ПФС находится на стоповом элементе.
- Хранимые действия остаются активными.
- Прекращается выполнение всей ПФС или ее части.

Если стоповый элемент находится в конце:	То:
последовательности	останавливается вся ПФС
ветви выбора	
пути в одновременной ветви	останавливается только этот путь, а остальная часть ПФС продолжает выполняться.

ПРИМЕР Использование стопового элемента

Когда ПФС доходит до шага last_step (последний шаг), и process_done (процесс выполнен) является истинным, выполнение ПФС останавливается.

Перезапуск (сброс) ПФС

Дойдя до стопового элемента, вы можете перезапустить ПФС несколькими способами:

Если ПФС является:	**И используется следующая опция *Last Scan of Active Steps* (Последнее сканирование активных шагов):**	**То:**
вложенной (т.е. другая ПФС вызывает данную ПФС как подпрограмму)	Automatic reset (Автоматический сброс)	В конце шага, вызывающего вложенную ПФС, происходит автоматический сброс вложенной ПФС: • Вложенная ПФС возвращается к начальному шагу. • Бит X стопового элемента во вложенной ПФС обнуляется.
	Programmatic reset (Программный сброс)	1. Используйте инструкцию SFC Reset (SFR) для перезапуска ПФС с требуемого шага. 2. Используйте логику для сброса бита X стопового элемента.
	Don't scan (Не сканировать)	
НЕ вложенной (т.е. *никакая* ПФС не вызывает данную ПФС как подпрограмму)	→	1. Используйте инструкцию SFC Reset (SFR) для перезапуска ПФС с требуемого шага. 2. Используйте логику для сброса бита X стопового элемента.

В следующем примере показано использование инструкции SFC Reset (SFR) для перезапуска ПФС и сброса бита X стопового элемента.

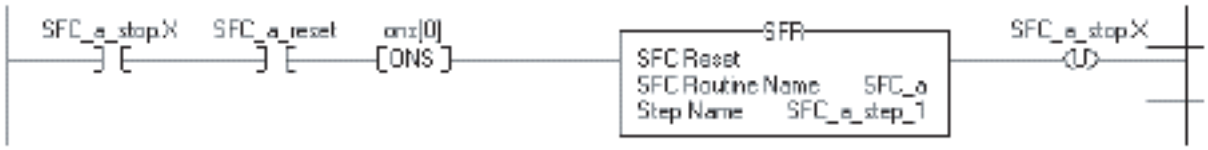
ПРИМЕР

Перезапуск (сброс) ПФС

Если SFC_a_stop.x = on (ПФС SFC_a находится на стоповом элементе) и SFC_a_reset = on (наступило время сброса ПФС), то для одного сканирования (ons[0] = on)произойдет следующее:

Возврат ПФС SFC_a к ее первому шагу SFC_a_Step_1

Обнуление бита X стопового элемента SFC_a_stop.X = 0



Структура SFC_STOP

Каждый стоп использует тег, предоставляющий следующую информацию о стоповом элементе:

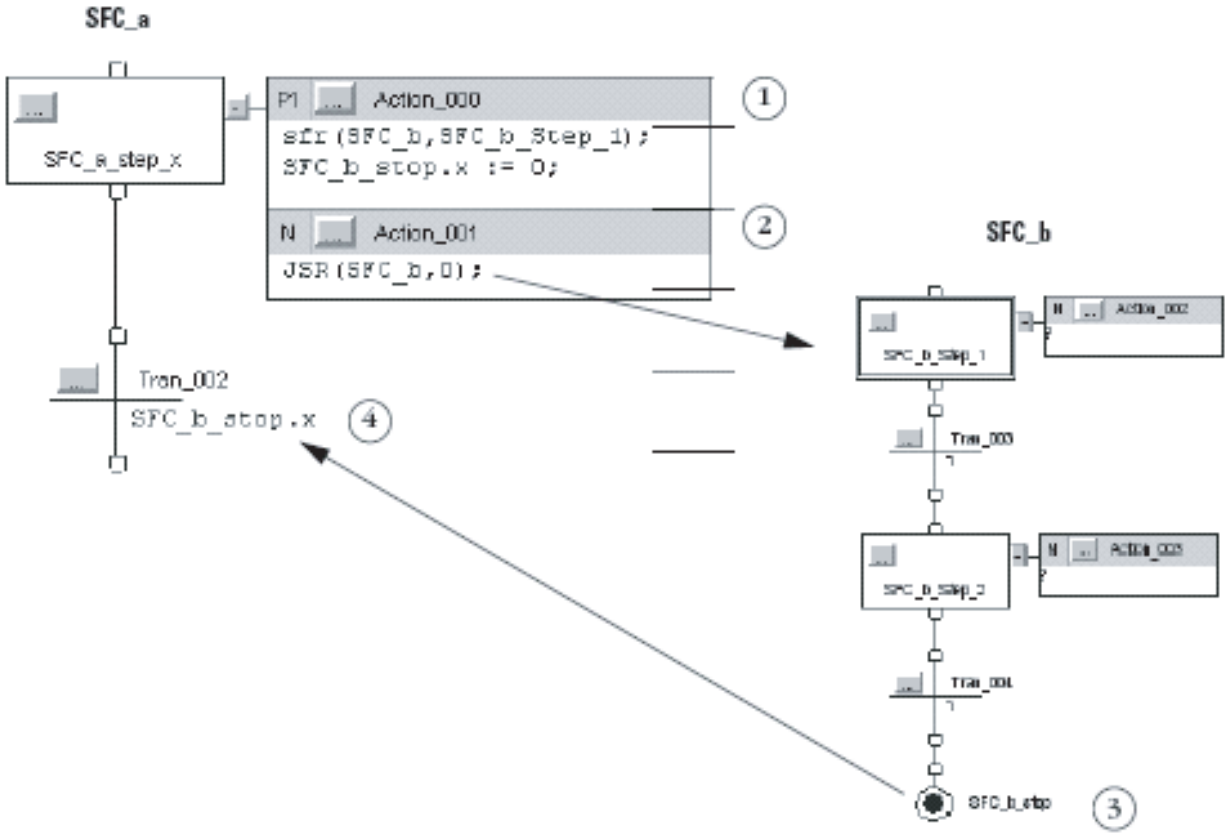
Если вы хотите:	То проверьте или установите этот член:	Тип данных:	Описание:
определить, когда ПФС дойдет до стопа	X	BOOL	• Когда ПФС доходит до стопа, устанавливается бит X. • Бит X обнулится при переходе контроллера из программного режима в режим выполнения, если вы сконфигурировали ПФС на перезапуск с начального шага. • Во вложенной ПФС бит X также сбрасывается при выходе из шага, вызывающего вложенную ПФС, если вы сконфигурировали ПФС на автоматический сброс.
определить место назначения инструкции SFC Reset (SFR)	Reset	BOOL	Инструкция SFC Reset (SFR) возвращает ПФС к заданному этой инструкцией шагу или стопу. • Бит Reset указывает, к какому шагу или стопу перейдет ПФС перед тем, как начать выполняться вновь. • При выполнении ПФС бит Reset сбрасывается.
определить, сколько раз становился активным стоповый элемент	Count	DINT	Это *не* является счетчиком сканирований стопового элемента. • Счетчик увеличивается на единицу всякий раз, когда стоповый элемент становится активным. • Он вновь увеличивается на единицу лишь после того, как стоповый элемент станет неактивным, а затем вновь станет активным. • Счетчик обнуляется лишь в том случае, если вы сконфигурировали ПФС на перезапуск с начального шага. При такой настройке он обнуляется, когда контроллер переходит из программного режима в режим выполнения.

Если вы хотите:	То проверьте или установите это член:	Тип данных:	Описание:	
использовать один тег для различных битов состояния данного стопового элемента	Status	DINT	**Для этого члена:**	**Используйте этот бит:**
			Reset	22
			X	31

Вложение ПФС

Одним из способов организации вашего проекта является создание одной ПФС, являющейся высокоуровневым представлением вашего процесса. Каждый шаг такой ПФС вызывает другую ПФС, выполняющую подробные процедуры данного шага (вложенную ПФС).

На следующем рисунке показан один из способов вложения ПФС. При таком способе для последнего сканирования ПФС используется опция *Programmatic reset* (Программный сброс) или *Don't scan* (Не сканировать). Если вы настроите ПФС на *Automatic reset* (Автоматический сброс), то шаг 1 в нижеприведенной последовательности действий выполнять не нужно.



1.. Сбросьте вложенную ПФС:

- Инструкция SFR перезапускает ПФС *SFC_b* с шага *SFC_b_Step_1*. При каждом выходе ПФС *SFC_a* из этого шага и последующем возврате вы должны сбросить *SFC_b*.
- Это действие также сбрасывает бит X стопового элемента.

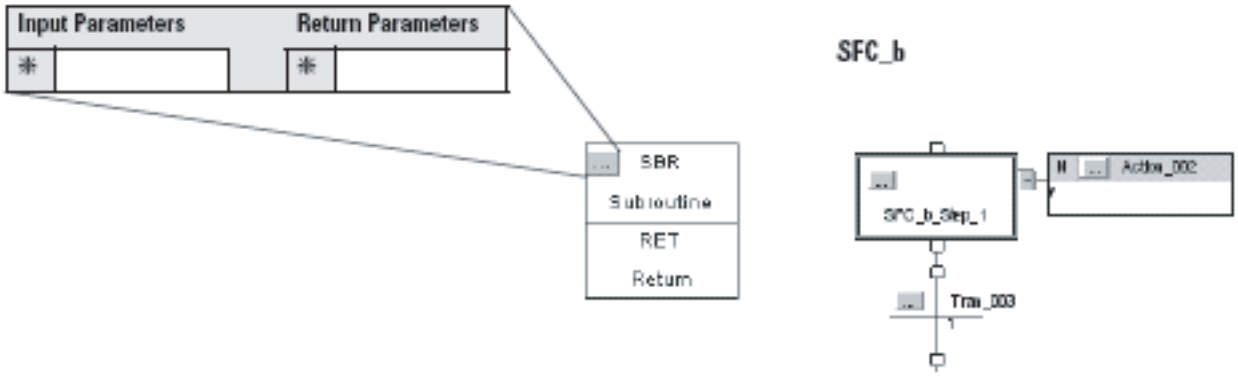
2. Вызовите ПФС *SFC_b*.

3. Остановите ПФС *SFC_b*. Это устанавливает бит X стопового элемента.

4. Используйте бит X стопового элемента, чтобы сигнализировать о том, что ПФС *SFC_b* выполнена и пора переходить к следующему шагу.

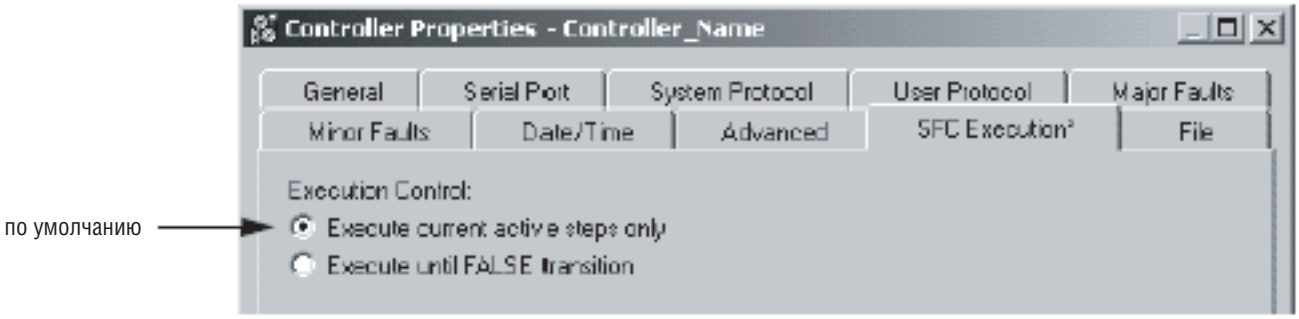
Передача параметров

Для передачи параметров в ПФС или из нее включите в ПФС элемент Subroutine/Return (Подпрограмма/Возврат).



Конфигурирование момента возврата к ОС/JSR

По умолчанию ПФС выполняет шаг или группу одновременных шагов и после этого возвращается к операционной системе (ОС) или вызывающей процедуре (JSR).



Вы можете использовать опцию выполнения ПФС до достижения ею перехода «ложно». Если одновременно несколько переходов истинны, то такой способ сокращает время перехода к требуемому шагу.

Используйте опцию *Execute until FALSE transition* (Выполнять до перехода ЛОЖНО) только в случае, если:

1. Вам не требуется обновлять параметры JSR перед каждым шагом. Параметры обновляются лишь при возврате ПФС к JSR. См. раздел «Передача параметров» на странице 5-50.
2. Переход «ложь» происходит в пределах времени сторожевого таймера для данной задачи. Если переход к JSR и выполнение оставшейся части задачи занимает больше времени, чем время сторожевого таймера, то происходит основная ошибка.

Подробная схема выполнения каждого варианта приводится на Рисунке 5.9 на странице 5-55.

Приостановка или сброс ПФС

Для расширенного управления выполнением вашей ПФС имеются две дополнительные инструкции:

Если вы хотите:	То используйте эту инструкцию:
приостановить ПФС	Pause SFC (SFP)
вернуть ПФС к определенному шагу или стопу	Reset SFC(SFR)

Обе эти инструкции имеются в языках релейной логики и структурированного текста.

За дополнительной информации обращайтесь к одному из следующих источников:

- Выберите *Instruction Help* (Справка по инструкциям) в меню *Help* (Справка) программного обеспечения RSLogix 5000. Посмотрите в категории *Program Control Instructions* (Инструкции программного управления).
- Посмотрите в Справочном руководстве по общим инструкциям для контроллеров Logix5000 (Logix5000 Controllers General Instructions Reference Manual), публикация 1756-RM003.

Схемы выполнения

Следующие схемы показывают выполнение ПФС с различной организацией шагов или различными опциями выполнения. Обращайтесь к этим схемам для получения более полного представления о выполнении ваших ПФС.

Рисунок 5.5 Выполнение последовательности

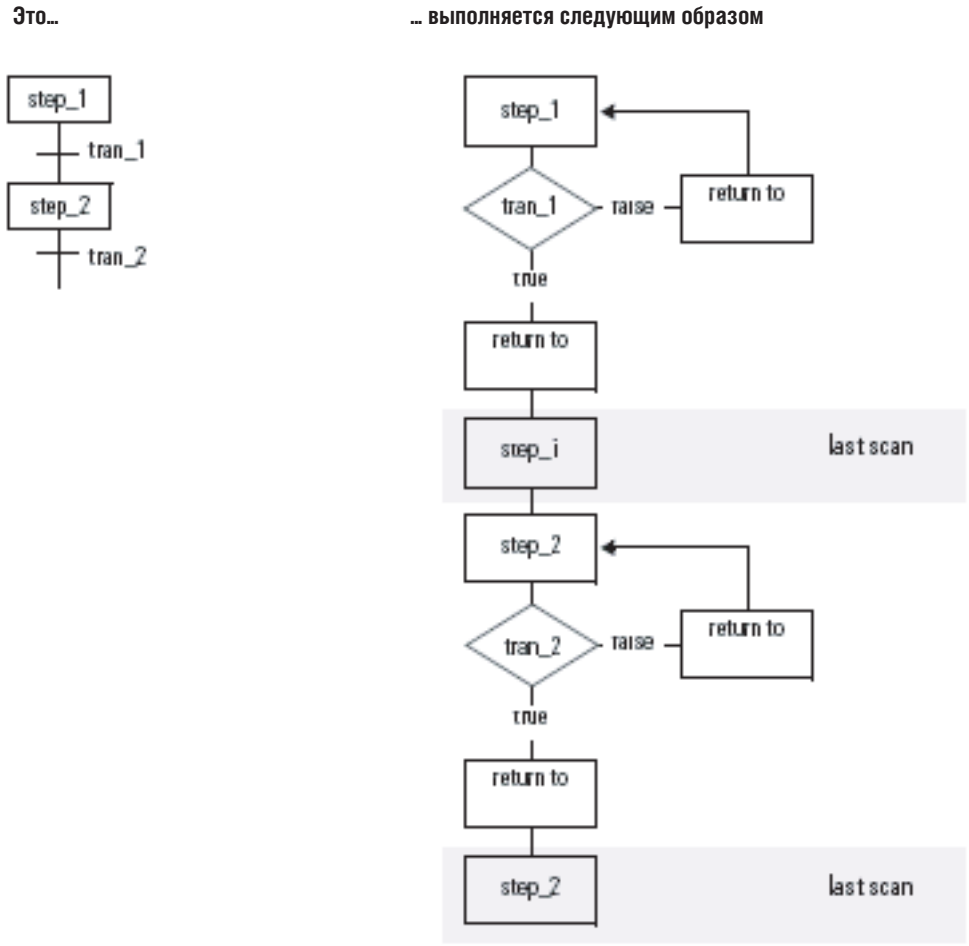
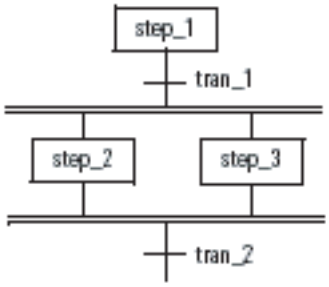


Рисунок 5.6 Выполнение одновременной ветви

Это...



... выполняется следующим образом

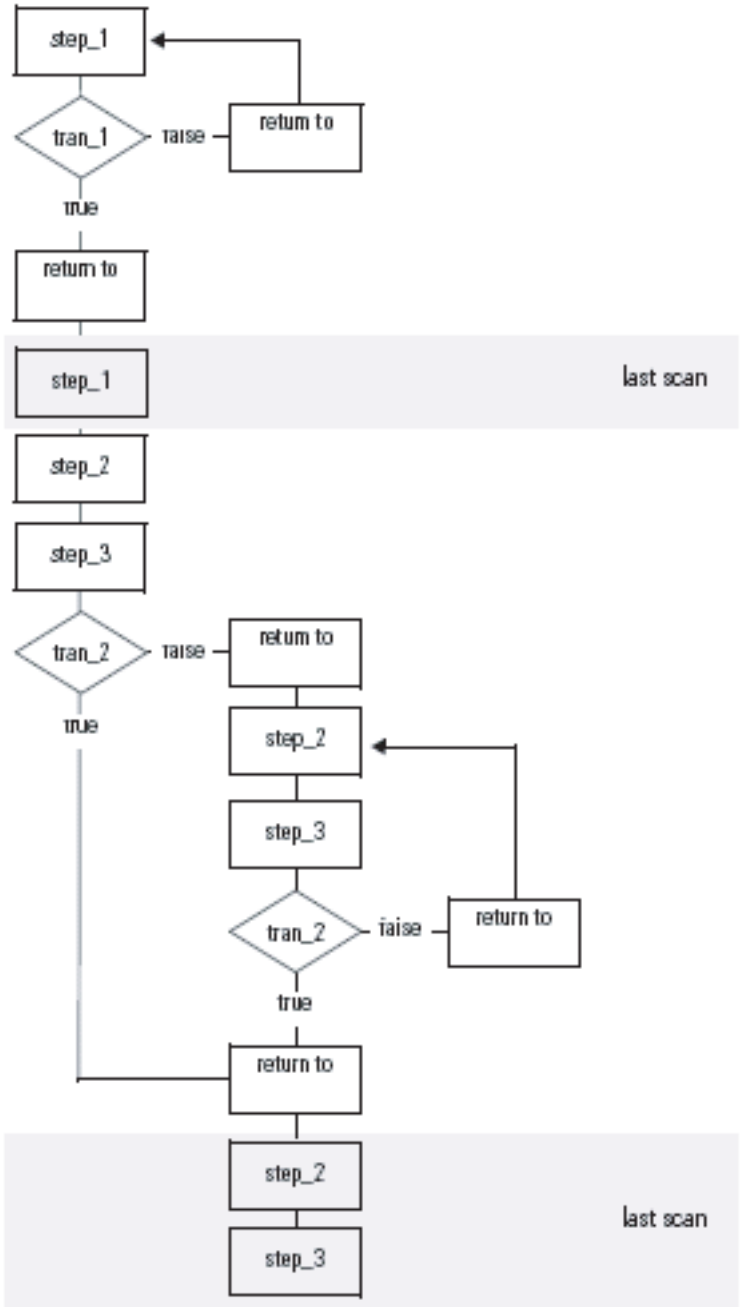


Рисунок 5.7 Выполнение ветви выбора

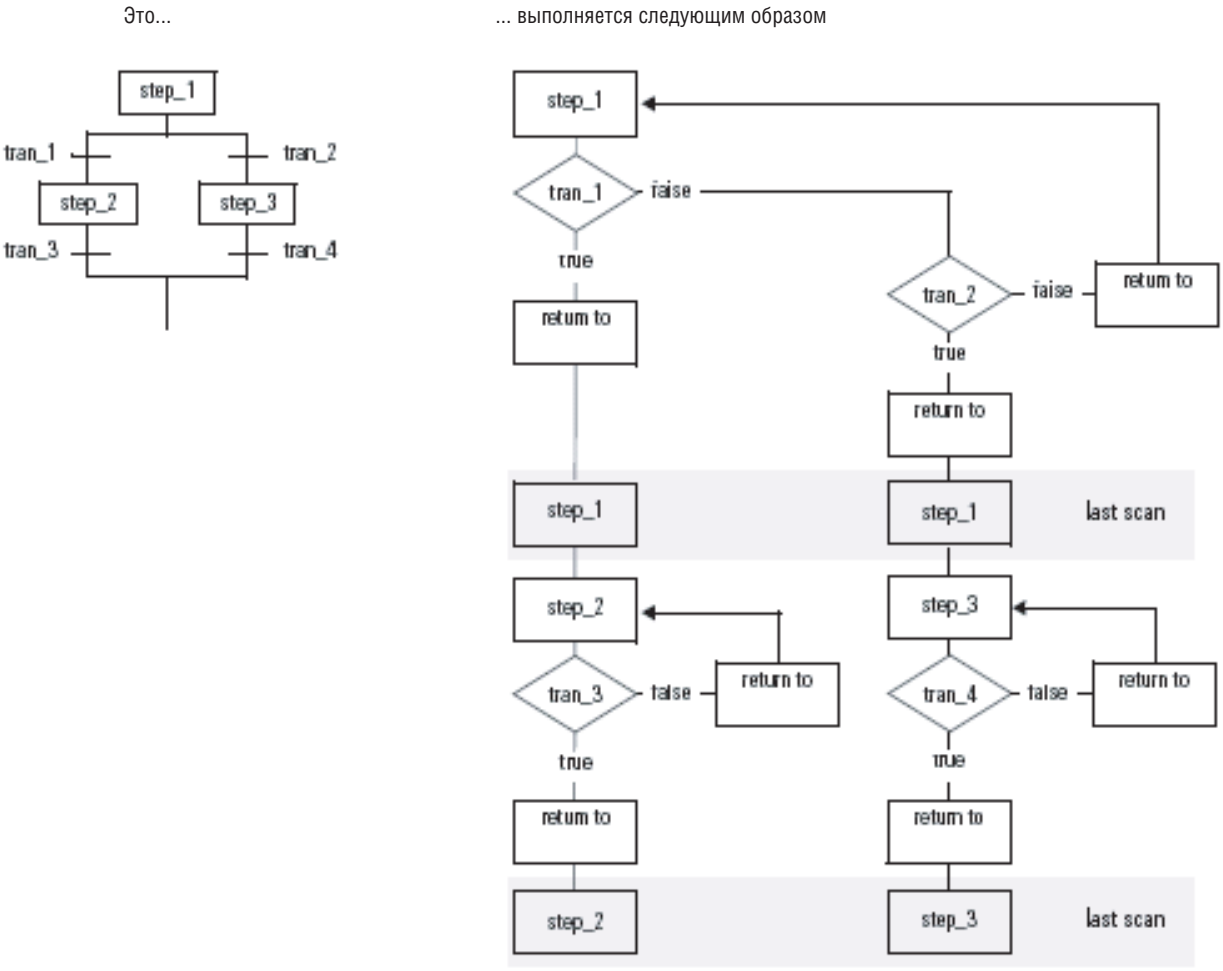


Рисунок 5.8 Вход/выход параметров в/из ПФС

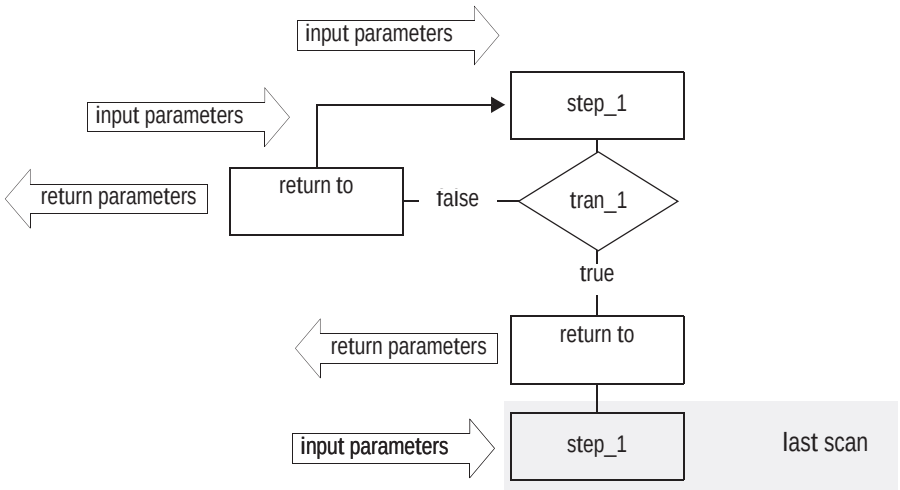
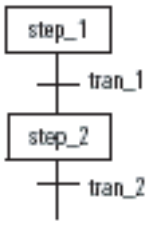


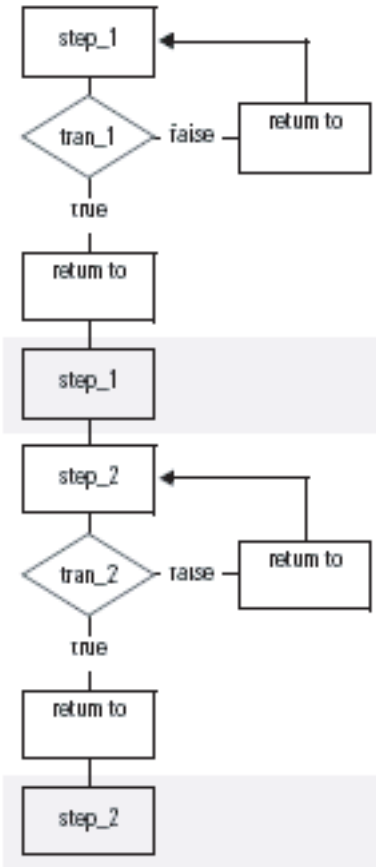
Рисунок 5.9 Варианты управления выполнением

Это...

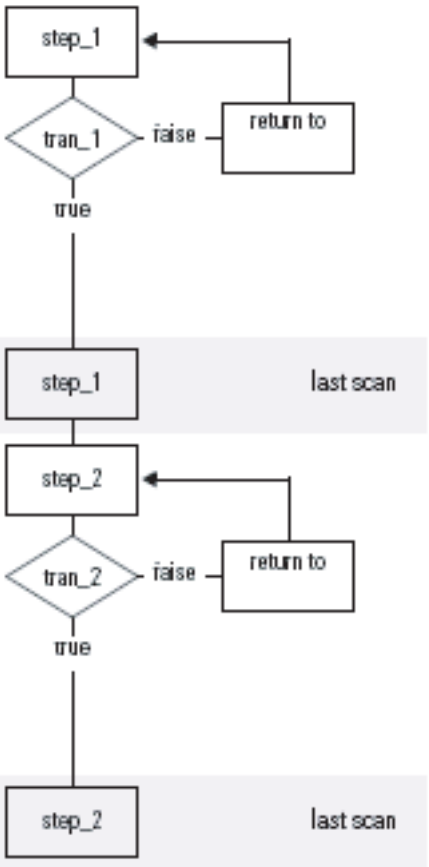


... выполняется следующим образом

Выполнять только активные в данный момент шаги



Выполнять до перехода ЛОЖНО



Для заметок:

Программирование последовательной функциональной схемы

Когда использовать данную процедуру

Используйте данную процедуру для ввода последовательной функциональной схемы (ПФС (SFC)) при работе в программной среде RSLogix 5000. Можно ввести уже созданную ПФС, а можно ее разработать и затем ввести. Процесс создания ПФС описан в разделе «Разработка ПФС» на стр. 5-1.

Перед началом использования данной процедуры

Перед началом использования данной процедуры убедитесь, что у вас есть разрешение на выполнение следующих задач:

- ☐ Navigate the Controller Organizer (Осуществлять навигацию контроллера)
- ☐ Identify the Programming Languages That Are Installed (Идентифицировать установленные языки программирования)

За более подробной информацией относительно данных задач обратитесь к разделу "Начало" (Getting Started) на стр. 1-1.

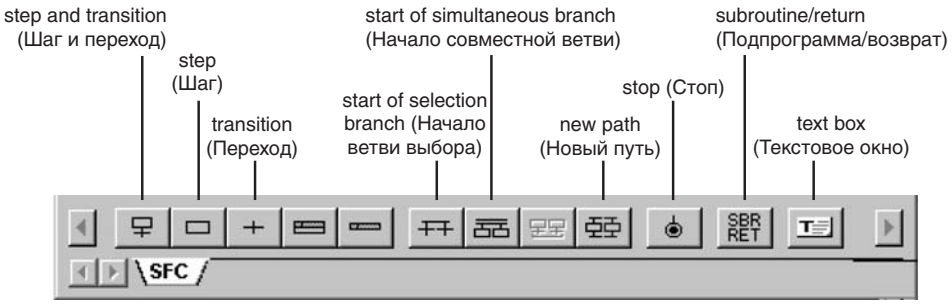
Как использовать данную процедуру

При программировании ПФС:

- ☐ Добавьте элемент SFC (ПФС) (Add an SFC Element)
- ☐ Создайте совместную ветвь (Create a Simultaneous Branch)
- ☐ Создайте ветвь выбора (Create a Selection Branch)
- ☐ Установите приоритеты для ветви выбора (Set the Priorities of a Selection Branch)
- ☐ Вернитесь на предыдущий шаг (Return to a Previous Step)
- ☐ Переименуйте шаг (Rename a Step)
- ☐ Сконфигурируйте шаг (Configure a Step)
- ☐ Переименуйте переход (Rename a Transition)
- ☐ Запрограммируйте переход (Program a Transition)
- ☐ Добавьте операцию (Add an Action)
- ☐ Переименуйте операцию (Rename an Action)
- ☐ Сконфигурируйте операцию (Configure an Action)
- ☐ Запрограммируйте операцию (Program an Action)
- ☐ Присвойте операциям порядок выполнения (Assign the Execution Order of Actions)
- ☐ Задокументируйте ПФС (Document the SFC)
- ☐ Разрешите вывод на экран или спрячьте текстовые окна или описания тегов (Show or Hide Text Boxes or Tag Descriptions)
- ☐ Сконфигурируйте выполнение ПФС (Configure the Execution of the SFC)
- ☐ Проверьте процедуру (Verify the Routine)

Добавление элемента ПФС (Add an SFC Element)

Для того чтобы добавить элементы ПФС используйте панель инструментов ПФС.



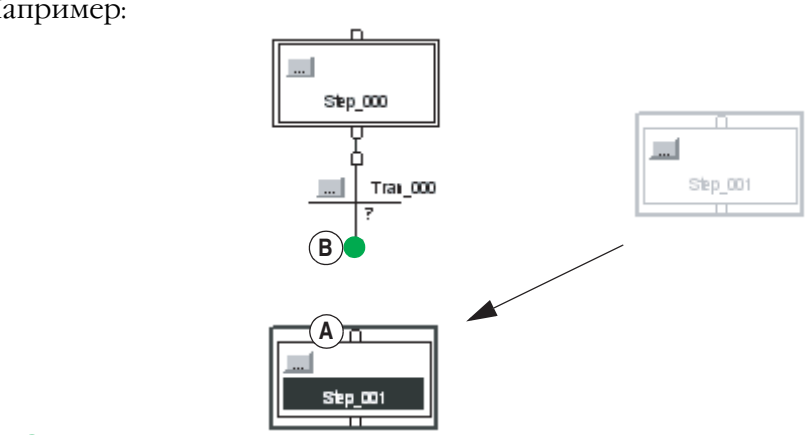
Для того, чтобы добавить элемент к вашей ПФС, у вас имеются следующие опции:

- ☐ Add and Manually Connect Elements (Добавить и вручную подсоединить элементы)
- ☐ Add and Automatically Connect Elements (Добавить и автоматически подсоединить элементы)
- ☐ Drag and Drop Elements (Добавить элементы методом буксировки)

Добавить и вручную подсоединить элементы

1. На панели инструментов ПФС щелкните на кнопке того элемента, который вы хотите добавить.
2. Отбуксируйте этот элемент в нужное место на ПФС.

   Например:



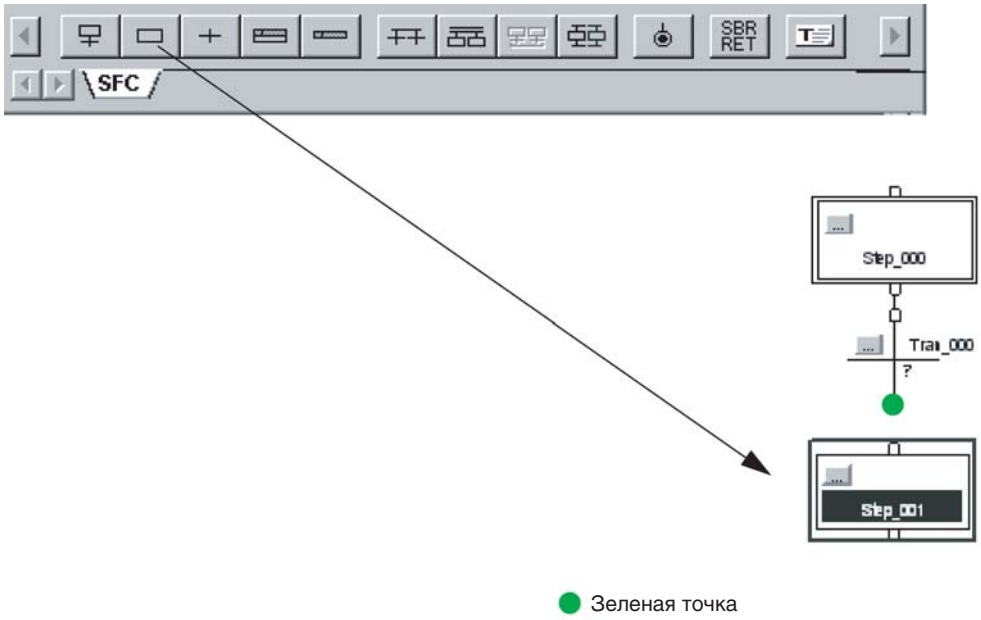
3. Соедините (скоммутируйте) два элемента друг с другом, щелкните на контакте элемента (A) , а затем на контакте другого элемента (B) . Зеленая точка указывает подходящее место контакта.

Добавить и автоматически подсоединить элементы

1. Выберите (щелчком) элемент, к которому вы хотите подсоединить новый элемент.
2. Пока данный элемент находится в режиме «выбран», на панели инструментов щелкните на кнопке для следующего элемента.

Добавить элементы методом буксировки

Из панели инструментов ПФС отбуксируйте кнопку требуемого элемента к нужной точке подключения на ПФС. Зеленая точка указывает подходящее место контакта.

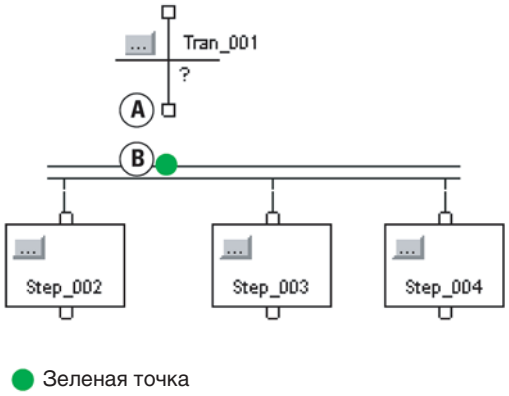


Создание совместной ветви

Начало совместной ветви

1. На панели инструментов ПФС щелкните на кнопке  . Отбуксируйте новую ветвь на нужное место.
2. Чтобы добавить путь к этой ветви, выберите (щелчком) первый шаг пути, который находится слева от того места, где вы хотите добавить новый путь.

   Затем щелкните на кнопке. 

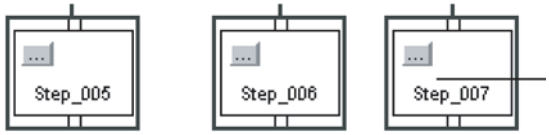


Зеленая точка

3. Чтобы подключить совместную ветвь к предыдущему переходу, щелкните на нижнем контакте этого перехода (A) , а затем щелкните на горизонтальной линии ветви (B) . Зеленая точка указывает подходящее место контакта.

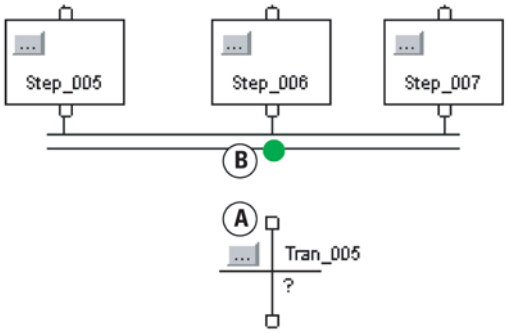
Окончание совместной ветви

1. Выберите последний шаг каждого пути в данной ветви. Чтобы выбрать эти пути вы можете:
   - либо щелкнуть и отбуксировать указатель по шагам, которые вы хотите выбрать,
   - либо, щелкнуть на первом шаге. Затем, нажав и удерживая [Shift] щелкнуть на оставшихся шагах, которые вы хотите выбрать.



2. Щелкните на кнопке  на панели инструментов ПФС.

3. Добавьте переход, который следует за совместной ветвью.



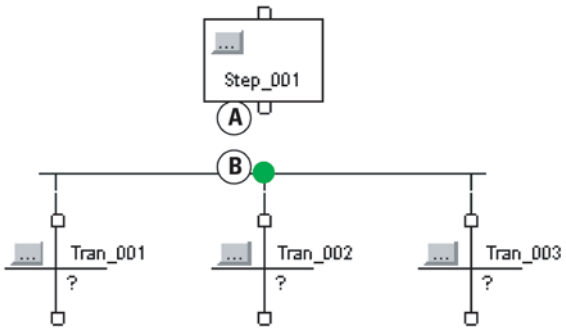
Зеленая точка

4. Чтобы подключить совместную ветвь к этому переходу, щелкните на верхнем контакте этого перехода (A) , а затем щелкните на горизонтальной линии этой ветви (B) . Зеленая точка указывает подходящее место контакта.

Создание ветви выбора

Начало ветви выбора

1. Щелкните на кнопке а  на панели инструментов ПФС. Затем отбуксируйте эту новую ветвь в желаемое место.
2. Чтобы добавить путь к этой ветви, выберите (щелчком) первый переход пути, который находится слева от того места, где вы хотите добавить новый путь. Затем щелкните на кнопке а  .

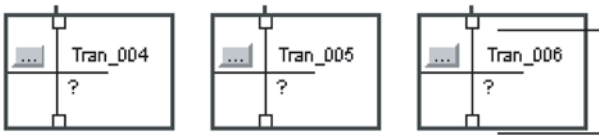


Зеленая точка

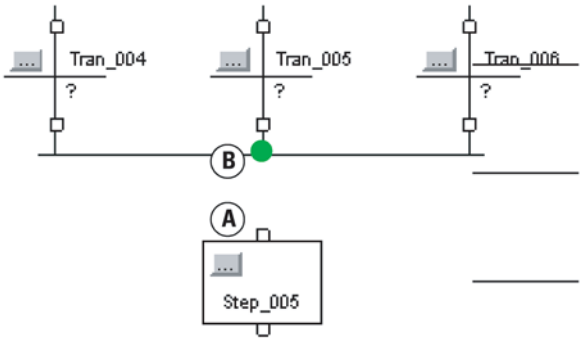
3. Чтобы подключить ветвь выбора к предыдущему шагу, щелкните на нижнем контакте этого шага (A) , а затем щелкните на горизонтальной линии ветви (B) . Зеленая точка указывает подходящее место контакта.

Окончание ветви выбора

1. Выберите последний переход каждого пути в данной ветви. Чтобы выбрать эти переходы, вы можете:
   - либо щелкнуть и отбуксировать указатель по переходам, которые вы хотите выбрать,
   - либо, щелкнуть на первом переходе. Затем, нажав и удерживая [Shift], щелкнуть на оставшихся переходах, которые вы хотите выбрать.



2. Щелкните на кнопке  на панели инструментов ПФС.
3. Добавьте шаг, который следует за ветвью выбора.



Зеленая точка

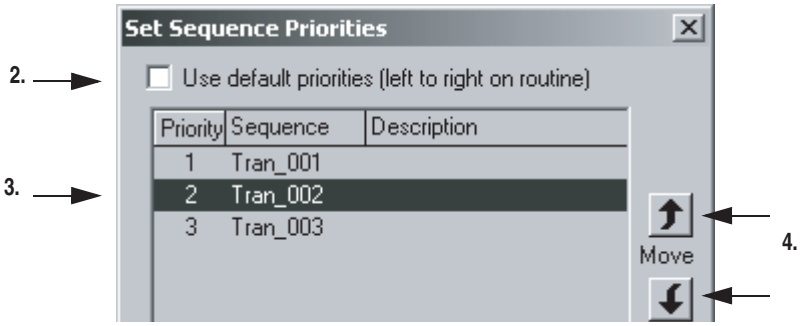
4. Чтобы подключить ветвь выбора к шагу, щелкните на верхнем контакте этого шага (A), а затем щелкните горизонтальной линии этой ветви (B). Зеленая точка указывает подходящее место контакта.

Настройки приоритетов ветви выбора

По умолчанию, ПФС проверяет переходы, которые начинают ветвь выбора слева направо. Если вы хотите проверить какой-либо другой переход первым, присвойте приоритеты каждому пути выбранной ветви. Например, хорошей практикой являет проверка, в первую очередь, условий ошибки. А затем, проверка обычных условий.

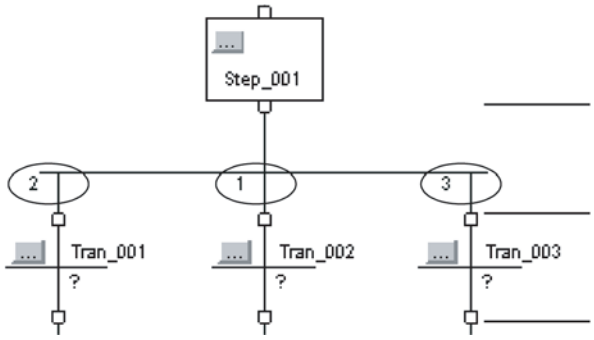
Чтобы присвоить приоритеты ветви выбора:

1. Щелкните правой клавишей мыши на горизонтальной линии, которой начинается данная ветвь и выберите *Set Sequence Priorities (Настройка последовательности приоритетов)*.



2. Уберите (снимите выбор) флажок *Use default priorities (Использовать приоритеты по умолчанию).*
3. Выберите переход.
4. Используйте кнопки перемещения *Move* для повышения или снижения приоритета данного перехода.
5. Когда для всех переходов установлены нужные приоритеты, выберите OK .

Когда вы снимаете флажок *Use default priorities* (Использовать приоритеты по умолчанию), цифры указывают приоритет каждого перехода.



Возвращение на предыдущий шаг

Чтобы перейти на другой шаг вашей ПФС:

- Подключите к шагу (Step) связь (Wire)
- Сделайте связь (Wire) невидимой
- Выведите на экран скрытую связь (Wire).

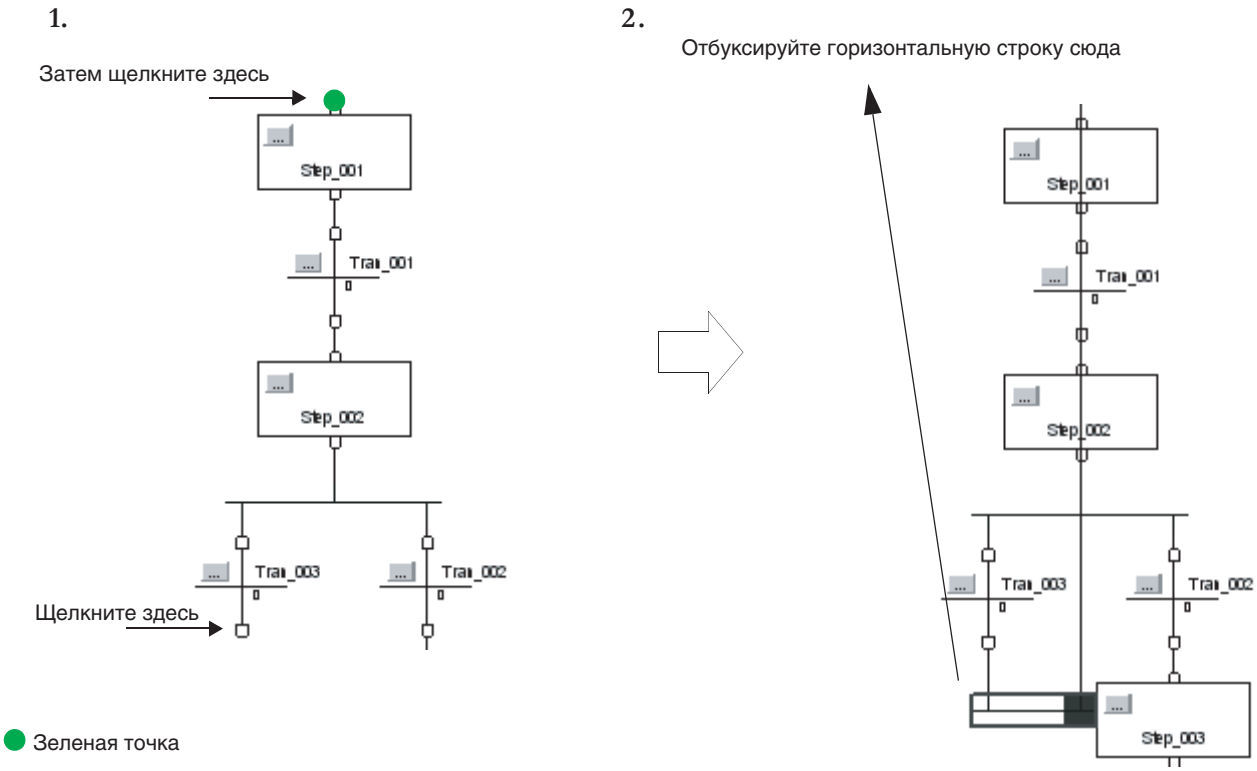
Подключение связи к шагу

1. Щелкните на нижнем контакте перехода, который сигнализирует о передаче управления. Затем щелкните на верхнем контакте шага, к которому вы хотите перейти. Зеленая точка указывает подходящее место контакта.

   Обычно, получаемая в результате связь, располагается посередине блок-схемы, и ее сложно разглядеть.

2. Чтобы сделать связь более заметной, переместите (буксировкой) ее горизонтальную строку в положение над шагом, к которому осуществляется переход. Возможно, вам понадобиться перекомпоновать отдельные элементы ПФС.

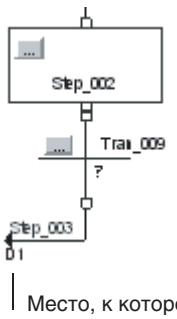
Например, переход к *Step_001* от *Tran_003*:



Скрытая связь

Если связь проходит через другие части вашей ПФС, то ее удобно сделать скрытой, чтобы облегчить просмотр схемы.

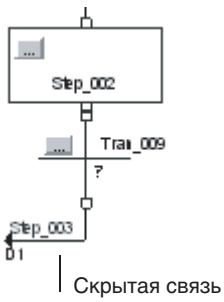
Чтобы спрятать связь, щелкните правой клавишей мыши на этой связи и выберите *Hide Wire (Спрятать связь)*.



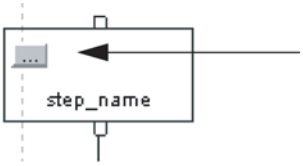
Чтобы увидеть элементы ПФС, к которым идет данная связь, щелкните в том месте, где располагается связь.

Вывод на экран скрытой связи

Чтобы вывести на экран скрытую связь, щелкните правой клавишей мыши на видимой части этой связи и выберите *Show Wire (Показать связь)*.

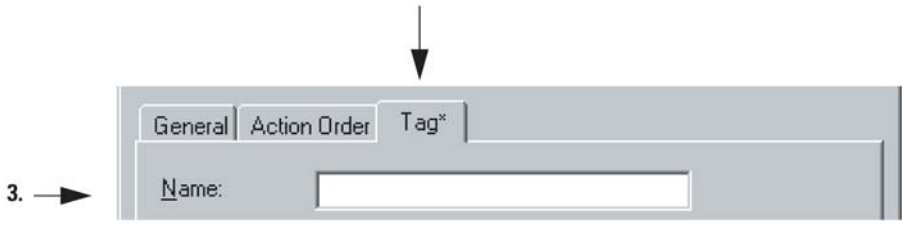


Изменение имени шага



Для каждого шага используется тег, в котором хранится информация о конфигурации и состоянии. Чтобы переименовать такой тег шага необходимо:

1. Щелкнуть на кнопке [...] этого шага.
2. Щелкнуть на закладке *Tag (Тег)*.



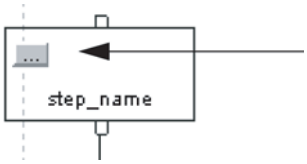
3. Ввести с клавиатуры новое имя для этого шага (тега).
4. Выбрать [OK] .

Конфигурирование шага

Для конфигурирования шага у вас имеются следующие опции:

- Assign the Preset Time for a Step (Присвоить шагу заданное время)
- Configure Alarms for a Step (Сконфигурировать для шага аварийные сигналы)
- Use an Expression to Calculate a Time (Использовать выражение для расчета времени).

Присвоение шагу заданного времени



1. Щелкните на кнопке [OK] этого шага.

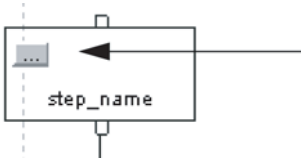
2. Введите с клавиатуры время для данного шага в миллисекундах.
3. Выберите OK.

Если данный шаг является активным в течение заданного времени, (Timer = Preset), устанавливается бит DN данного шага.

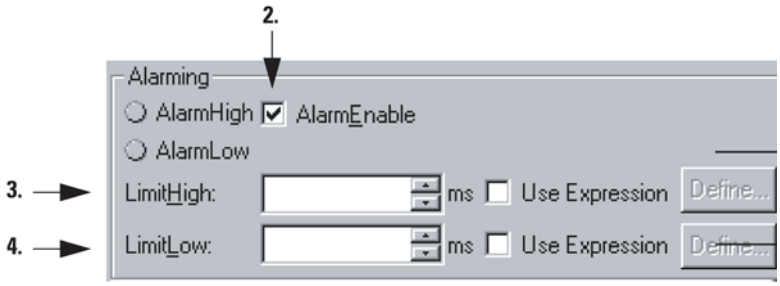
Для того чтобы узнать, как рассчитать заданное время для шага в режиме выполнения, обратитесь к разделу “Использование выражения для расчета времени» на стр. 6-12.

Конфигурирование сигналов тревоги для шага

Чтобы включить сигнал тревоги, если шаг выполняется слишком долго или является недостаточно продолжительным:



1. Щелкните на кнопке … заданного шага.
2. Установите флажок *AlarmEnable (Сигнализация)*.



3. Введите с клавиатуры верхнее значение времени для сигнала тревоги в миллисекундах.
4. Введите с клавиатуры нижнее значение времени для сигнала тревоги в миллисекундах.
5. Выберите OK .

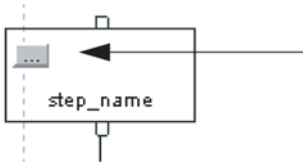
Для того чтобы узнать, как рассчитать заданное время для сигнала тревоги в режиме выполнения, обратитесь к разделу “Использование выражения для расчета времени» на стр. 6-12.

Использование выражения для расчета времени

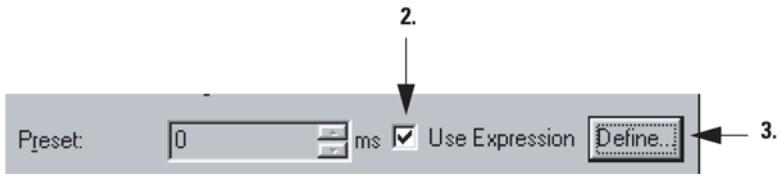
Для расчета времени на основе тегов в вашем проекте, введите числовое выражение. Вы можете использовать выражения для расчета следующих значений:

- Preset (Заданное время)
- LimitHigh (Верхний предел)
- LimitLow (Нижний предел)

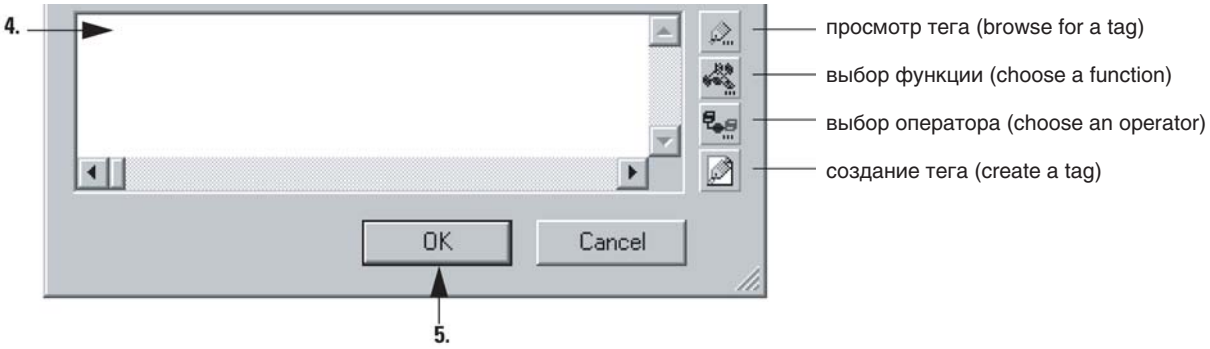
Чтобы ввести выражение для времени:



1. Щелкните на кнопке … данного шага.
2. Выберите (установите) флажок *Use Expression (Использовать выражение)*.



3. Щелкните на кнопке *Define (Определить)*.



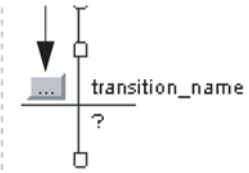
4. Введите с клавиатуры **числовое выражение**, которое определит данное время.

- Для облегчения ввода выражения используйте кнопки сбоку диалогового окна.
- За дополнительной информацией обратитесь к разделу «Выражения» на стр. 7-4.

5. Выберите OK .

6. Чтобы закрыть диалоговое окно *Step Properties (Свойства шага),* выберите OK .

Переименование перехода



Для каждого перехода используется тег с информацией о состоянии данного перехода. Чтобы переименовать тег перехода:

1. Щелкните на кнопке [...] данного перехода.
2. Щелкните на закладке *Tag (Тег)*.

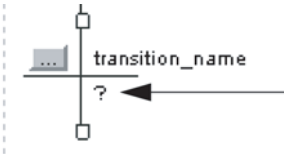
3. Введите с клавиатуры новое имя для данного перехода (тега).
4. Выберите [OK] .

Программирование перехода

Для программирования перехода вы можете использовать следующие опции:

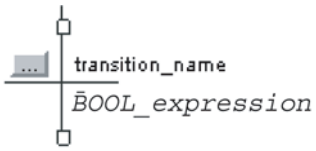
- Enter a BOOL Expression (Ввод булева выражения)
- Call a Subroutine (Вызов подпрограммы)

Ввод булева выражения



Одним из самых простых способов программирования перехода является ввод условий в качестве булева выражения на языке структурированного текста. За дополнительной информацией о булевых выражениях обратитесь к разделу «Выражения» на стр. 7-4.

1. Дважды щелкните в текстовой области данного перехода.
2. Введите с клавиатуры булево выражение, определяющее, когда переход имеет значения «истина» или «ложь».
3. Чтобы закрыть окно текстового ввода, нажмите [Ctrl] + [Enter].

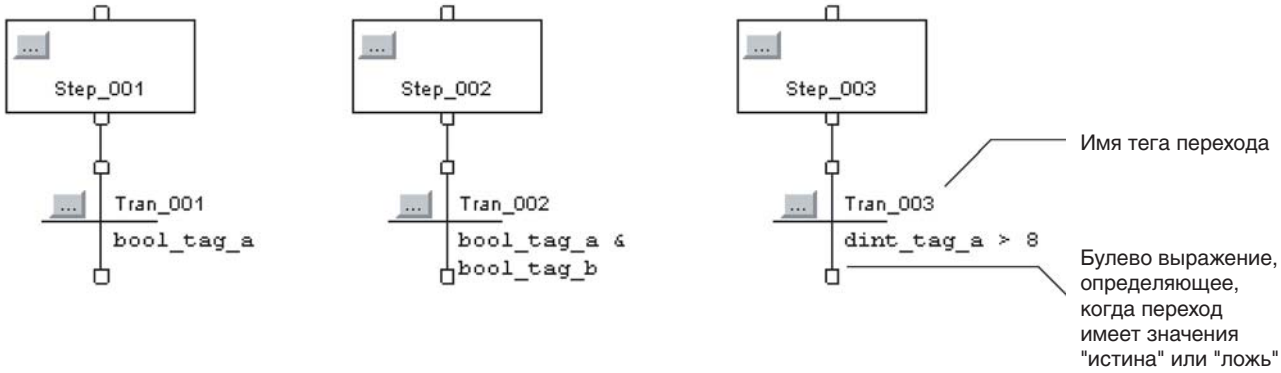


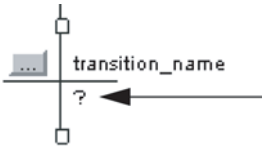
В примере, следующем ниже, представлены три перехода, использующих булевы выражения.

.

ПРИМЕР

Ведите булево выражение

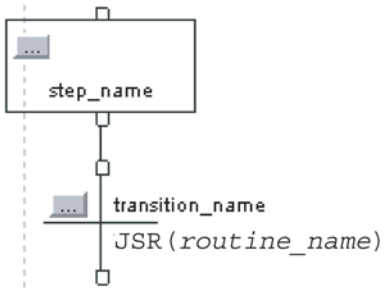




Вызов подпрограммы

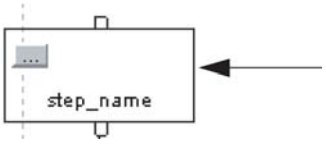
1. Щелкните правой клавишей мыши на переходе и выберите *Set JSR (Настройка JSR)*.

2. Выберите процедуру, которая содержит алгоритм для данного перехода.
3. Выберите OK .

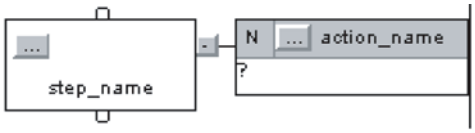


Добавление операции

Чтобы в рамках шага добавить операцию:

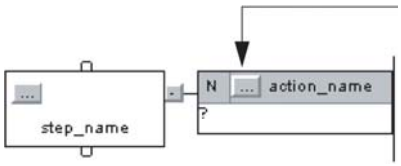


Щелкните правой клавишей мыши на шаге, в котором выполняется операция, и выберите *Add Action (Добавить операцию)*.



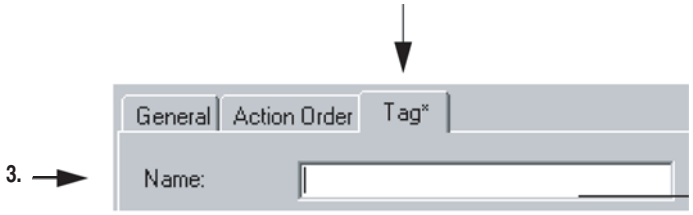
Переименование операции

Чтобы изменить имя операции на другое, подходящее для вашего приложения:



1. Щелкните на кнопке [...] данной операции.

2. Щелкните на закладке *Tag (Тег)*.



3. Введите с клавиатуры новое имя данной операции (тега).

4. Выберите [OK] .

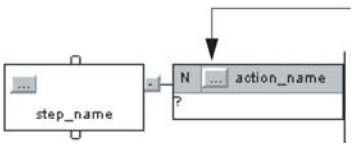
Конфигурирование операции

Для конфигурирования операции вы можете использовать следующие опции:

- Change the Qualifier of an Action (Изменить управляющий параметр опции)
- Calculate a Preset Time at Runtime (Рассчитать заданное время при выполнении)
- Mark an Action as a Boolean Action (Обозначить операцию как булеву операцию)

Изменение управляющего параметра операции

Управляющий параметр операции определяет, когда операция начинается и заканчивается. Значение управляющего параметра по умолчанию - *N Non-Stored (не сохраняется)*. Операция начинается, когда активируется шаг и заканчивается при завершении шага. За более подробной информацией обратитесь к разделу «Выбор управляющего параметра операции» на стр. 5-23.



1. Щелкните на кнопке [...] данной операции.

2. Присвойте данной операции управляющий параметр.

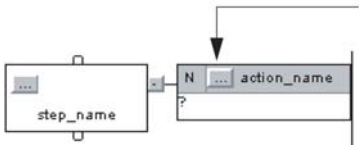
3. Если вы выбрали временной параметр, введите с клавиатуры временную границу или задержку для данной операции в миллисекундах. Параметры, связанные со временем, включают в себя:

- L Time Limited (Ограниченные по времени)
- SL Stored and Time Limited (Сохраняемые и ограниченные по времени)
- D Time Delayed (С задержкой по времени)
- DS Delayed and Stored (С задержкой по времени и сохраняемые)
- SD Stored and Time Delayed (Сохраняемые и с задержкой по времени)

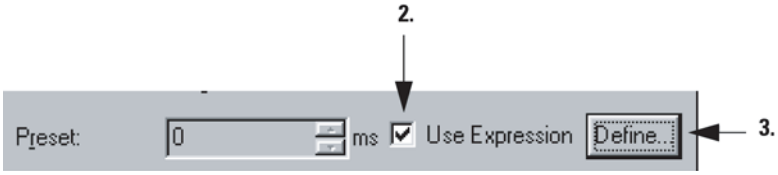
4. Выберите [OK] .

Расчет заданного времени при выполнении

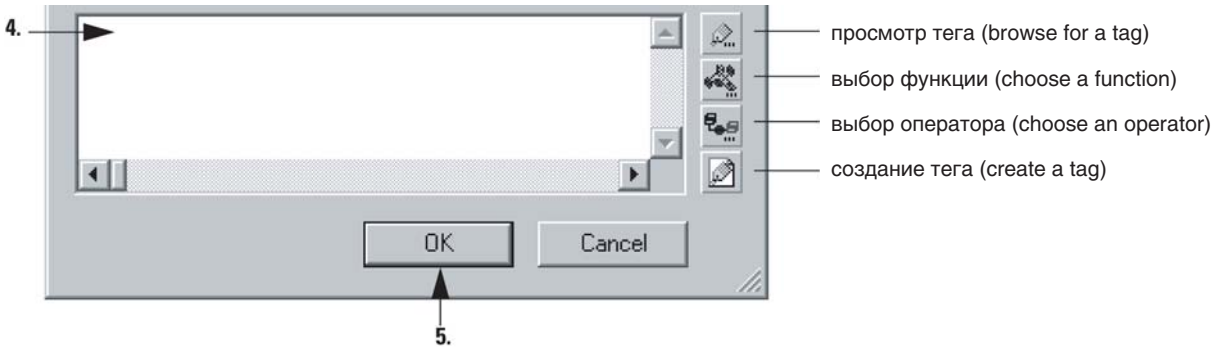
Для расчета заданного времени на основе тегов в вашем проекте, введите значение в виде **числового выражения**.



1. Щелкните на кнопке [...] данной операции.
2. Выберите (установите) флажок *Use Expression (Использовать выражение)*.



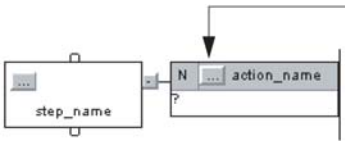
3. Щелкните на кнопке *Define (Определить)*.



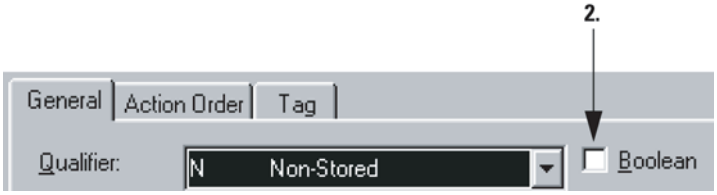
4. Введите с клавиатуры **числовое выражение**, которое определит заданное время.
   - Для облегчения ввода выражения используйте кнопки сбоку диалогового окна.
   - За дополнительной информацией обратитесь к разделу «Выражения» на стр. 7-4.
5. Выберите [OK] .
6. Чтобы закрыть диалоговое окно *Action Properties (Свойства операции)*, выберите [OK] .

Обозначение операции как булевой операции

Используйте булеву операцию только для установки бита при выполнении операции. За более подробной информацией обратитесь к разделу «Использование булевых операций» на стр. 5-20.



1. Щелкните на кнопке [...] данной операции.



2. Щелкните на флажке *Boolean (Булевы)*.

3. Выберите [OK] .

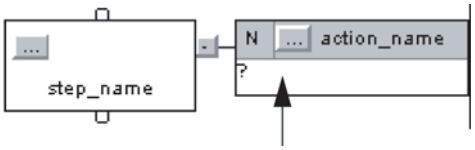
Программирование операции

Для программирования операции вы можете использовать следующие опции:

- Enter Structured Text (Ввод структурированного текста)
- Call a Subroutine (Вызов подпрограммы)

Ввод структурированного текста

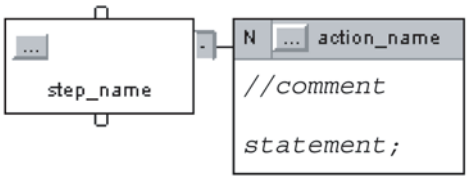
Одним из самых простых способов программирования операции является запись алгоритма в качестве структурированного текста в пределах данной операции. Когда операция включается, контроллер выполняет структурированный текст.



1. Дважды щелкните в текстовой области данной операции.

2. Введите с клавиатуры структурированный текст.

3. Чтобы закрыть окно текстового ввода, нажмите [Ctrl] + [Enter].

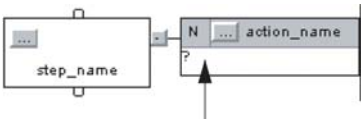


За более подробной информацией обращайтесь:

За данной информацией:	**Обращайтесь к:**
Общая информация о присваивании, операторах, функциях, инструкциях, комментариях	«Программирование на языке структурированного текста» на стр. 7-1
Отдельные инструкции	• *Logix5000 Controllers General Instructions ReferenceManual*, publication 1756-RM003 Инструкции для контроллеров *Logix5000 справочное руководство*, документ 1756-RM003 • Logix5000 Controllers Process and Drives Instructions Инструкции по обработке и управлению для контроллеров Logix5000 *справочное руководство*, документ 1756-RM006 • Logix5000 Controllers Motion Instruction Set Инструкции по контролю движения для контроллеров Logix5000 *справочное руководство*, документ 1756-RM007

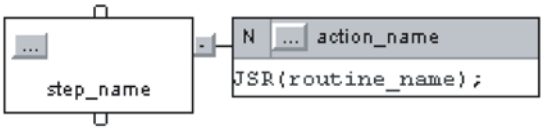
Вызов подпрограммы

Для выполнения подпрограммы при активной операции используйте инструкцию Jump to Subroutine (JSR) (Переход к подпрограмме).



1. В области ввода текста операции ПФС щелкните правой клавишей мыши и выберите *Set JSR (Настройка JSR)* .

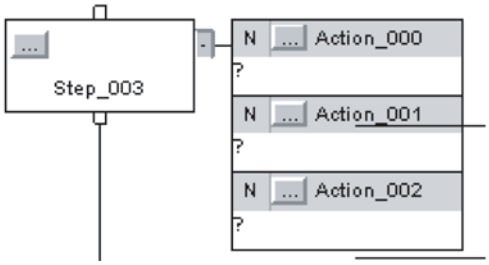
2. Выберите процедуру, которую вы хотите вызвать.
3. Для передачи параметра в процедуру щелкните на пустом окне *Input Parameters (Ввод параметров).* Затем используйте стрелку вниз для выбора тега, содержащего этот параметр.
4. Для получения параметра от процедуры, щелкните на пустом окне *Return Parameters (возврат параметров).* Затем используйте стрелку вниз для выбора тега, сохраняющего содержащего этот параметр из процедуры.
5. Выберите OK .



Присваивание порядка выполнения операции

Операции выполняются в порядке их следования.

Например:



Когда активен Step_003, его операции выполняются в следующей последовательности:

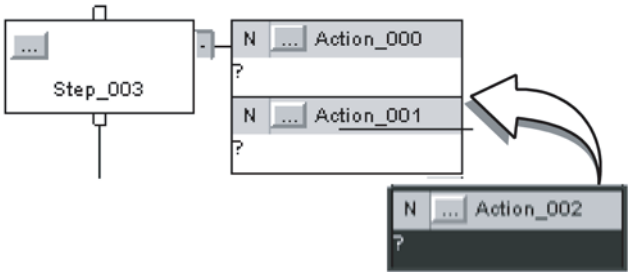
1. Action_000

2. Action_001

3. Action_002

Чтобы изменить порядок выполнения какой-либо операции, отбуксируйте эту операцию на желаемое место в последовательности операций. Зеленая линейка указывает место размещения.

Например:



Документирование ПФС

Для документирования ПФС у вас имеются следующие опции:

Для документирования информации о:	И вы хотите:	Сделайте это:
общей информации о ПФС	→	Добавьте текстовое окно
шаге	→	Добавьте текстовое окно Или Добавьте описание тегов
переходе	Загрузить документацию в контроллер	Добавьте комментарии на языке структурированного текста
	Имеете опцию для того, чтобы вывести на экран или спрятать документацию	Добавьте текстовое окно
	Разместить документацию где-либо в ПФС	Или Добавьте описание тегов
операции	Загрузить документацию в контроллер	Добавьте комментарии на языке структурированного текста
стоп	→	Добавьте текстовое окно
Другом элементе (напр. ветви выбора)	→	Или Добавьте описание тегов

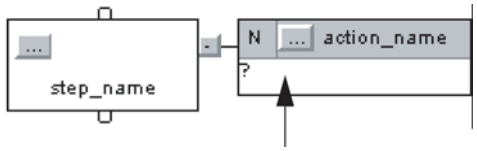
Добавить комментарии на языке структурированного текста

Для форматирования комментариев используйте следующую таблицу:

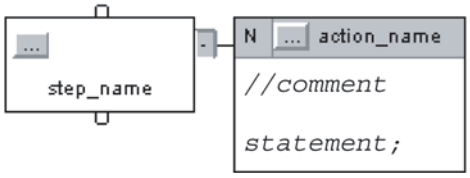
Чтобы добавить комментарий:	Используйте один из следующих форматов:
На одной строке	`//comment`
В конце строки структурированного текста	`(*comment*)` `/*comment*/`
В пределах строки структурированного текста	`(*comment*)` `/*comment*/`
Если занимает больше одной строки	`(*start of comment . . . end of comment*)` `/*start of comment . . . end of comment*/`

За более подробной информацией обратитесь к разделу «Комментарии» на стр. 7-28.

Для ввода комментария:

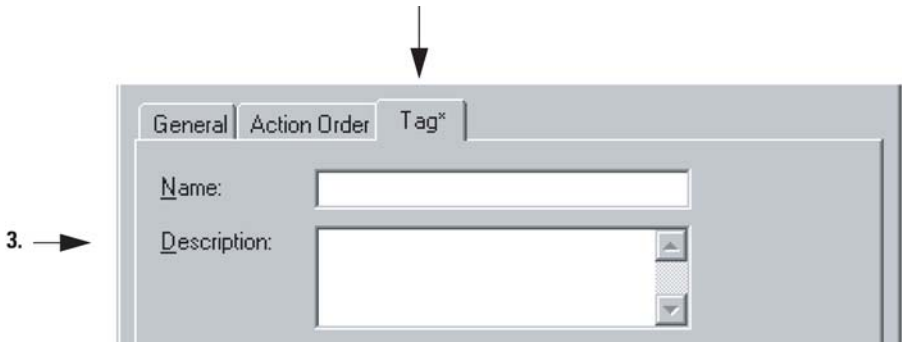


1. Дважды щелкните в текстовой области данной операции.
2. Введите с клавиатуры комментарии.
3. Чтобы закрыть окно текстового ввода, нажмите [Ctrl] + [Enter].



Добавить описания тега

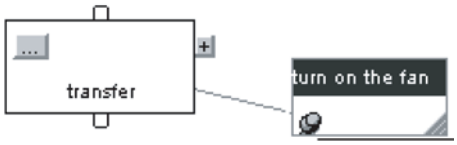
1. Щелкните на кнопке [...] данного элемента.
2. Щелкните на закладке *Tag (Тег)*.



3. Введите с клавиатуры описание для данного элемента (тега).
4. Выберите [OK] .
5. Отбуксируйте окно с описанием на желаемое место в ПФС.

Добавить текстовое окно

Текстовое окно позволяет вам добавлять комментарии, поясняющие работу элементов ПФС (шага, перехода, стоп, и т.д.). Текстовое окно может использоваться для ввода информации, которая будет использована вами позднее. Например:

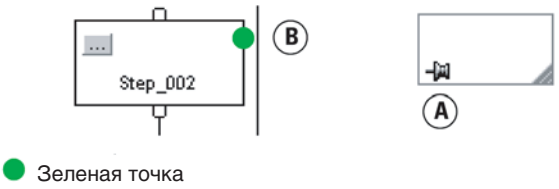


1. Щелкните на 

   Появится текстовое окно. 

2. Отбуксируйте это текстовое окно на место около элемента, к которому оно относится.
3. Дважды щелкните в текстовом окне и введите с клавиатуры текст. Нажмите [Ctrl] + [Enter].
4. Если вы перемещаете элемент в ПФС, то чтобы вам хотелось чтобы происходило с текстовым окном?

Если текстовое окно:	То:
Остается в той же точке	Stop. Вы все сделали.
Перемещается вместе с элементом, к которому оно относится	Переходите к пункту 5.



5. Щелкните на символе контакта в этом текстовом окне, а затем щелкните на элементе ПФС, к которому вы хотите привязать это текстовое окно. Зеленая точка указывает подходящее место присоединения.

Показать или спрятать текстовое окно или описание тегов

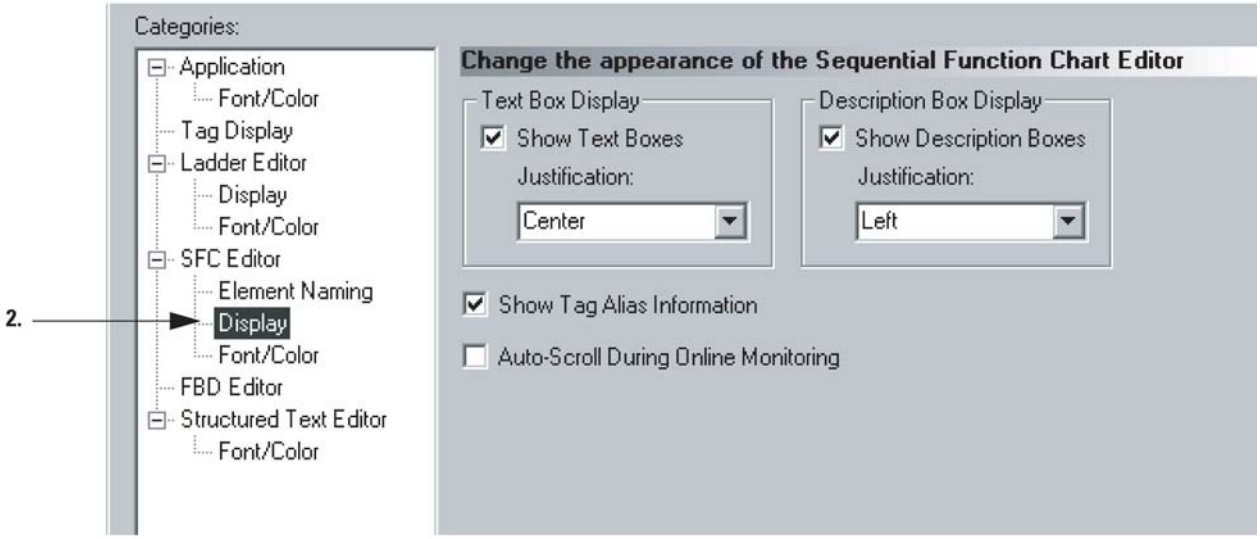
В вашем распоряжении имеются опции, которые позволяют показать или спрятать текстовые окна или описания тегов. Если вы выбрали показ описаний, ПФС показывает описания только для шагов, переходов, останова (а не операций).

Чтобы показать или спрятать текстовые окна или описания у вас имеются следующие опции:

- Show or Hide Text Boxes or Descriptions (Показать или спрятать текстовые окна или описания)
- Hide an Individual Tag Description (Спрятать описание отдельного тега)

Показать или спрятать текстовые окна или описания

1. В меню *Tools (Инструменты)* выберите *Options (Опции)*.



2. В *SFC Editor (Редакторе ПФС)* выберите раздел *Display (Дисплей)*.

3. Выберите нужную опцию.

Если вы хотите:	То:
Показать текстовые окна или описания	Установите соответствующий флажок
Спрятать текстовые окна или описания	Снимите соответствующий флажок

4. Выберите OK .

Спрятать описание отдельного тега

Чтобы спрятать описание заданного элемента при просмотре других описаний:

1. Щелкните на кнопке [...] элемента, описание которого вы хотите спрятать.
2. Установите флажок *Never display description in routine (Никогда не показывать описание в процедуре).*

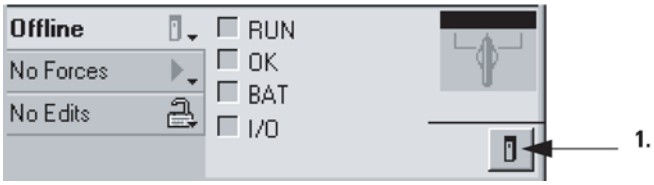
3. Выберите [OK] .

За информацией о показе других описаний, обратитесь к разделу «Показать или спрятать текстовые окна или описания» на стр. 6-26.

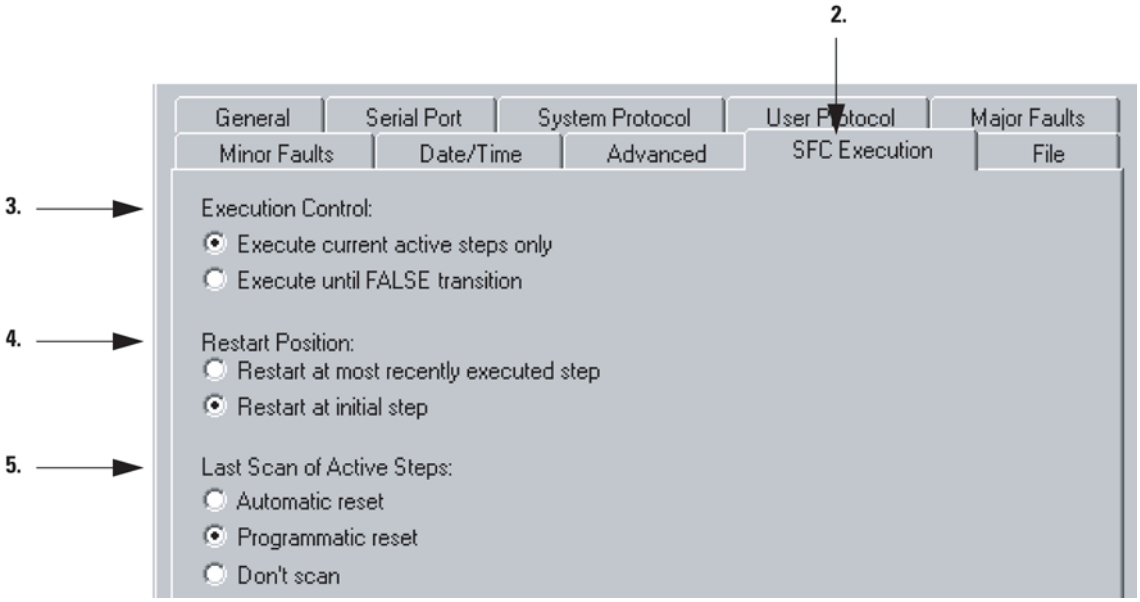
Конфигурирование выполнения ПФС

Закладка SFC Execution (Выполнение ПФС) окна свойств контроллера позволяет настраивать:

- Что делать, если переход имеет значение «истина»,
- Где начинать после перехода в режим выполнения или восстановления после потери питания,
- Что делать на последнем сканировании шага.



1. На панели инструментов Online щелкните на кнопке свойств контроллера.
2. Щелкните на закладке *SFC Execution (Выполнение ПФС)*.



3. Выберите, возвращать или нет OS/JSR если переход имеет значение «истина».
4. Выберите где перезапустить ПФС после перехода в режим выполнения или восстановления после потери питания.
5. Выберите, что делать на последнем сканировании шага.
6. Выберите OK .

Проверка процедуры

По мере программирования процедуры, периодически проверяйте свою работу:

1. На панели инструментов окна RSLogix 5000 щелкните на  .

2. Если внизу окна будет перечислены какие-либо ошибки:

a. Перейдите на первую ошибку или предупреждение и нажмите [F4].

b. Исправьте ошибку в соответствии с описанием в окне Results (Результаты).

c. Перейдите к пункту 1.

3. Чтобы закрыть окно Results (Результаты), нажмите [Alt] + [1].

Для заметок:

Для заметок:

Программирование на языке структурированного текста

Когда пользоваться данной главой

Эта глава полезна при вводе структурированного текста в:

- Процедурах,
- Операциях последовательных функциональных схем (ПФС),
- Переходах последовательных функциональных схем (ПФС).

Синтаксис языка структурированногm текста

Структурированный текст является текстовым языком программирования, использующим операторы для задания действий для выполнения.

- Операторы структурированного текста не чувствительны к регистру.
- Использование символов табуляции и возврата каретки (отдельные строки) делает программу легкой для чтения.

Структурированный текст может содержать следующие элементы:

Термин:	Определение:		Примеры:
присваивание (см. стр. 7-2)	Используйте оператор присваивания для присвоения значений тегам. Символом присваивания является «:=». Присваивание заканчивается точкой с запятой «;».		`tag := expression;`
выражение (см. стр. 7-4)	Выражение является частью таких элементов, как присваивание и конструкция. Выражение работает с числами (числовое выражение) или логическими величинами (логическое выражение (BOOL)). Выражение содержит:		
	Теги:	Поименованную область памяти для хранения данных (типа BOOL,SINT,INT,DINT, REAL, строка).	`value1`
	Непосредственные значения:	Постоянные значения.	`4`
	Операторы:	Символ или мнемоника, задающие операцию в пределах выражения.	`tag1 + tag2` `tag1 >= value1`
	Функции:	При выполнении функция выдает одно значение. Операнд функции помещается в круглые скобки. Даже при одинаковом синтаксисе функции отличаются от инструкций, в выражениях используются только функции. Инструкции не могут использоваться в выражениях.	`function(tag1)`

Термин:	Определение:	Примеры:
инструкция (см. стр. 7-11)	Инструкция является отдельно расположенным оператором. Операнды инструкции помещаются в круглые скобки. В зависимости от типа инструкции, она может не содержать операндов, содержать один, два или несколько операндов. При выполнении инструкция имеет на выходе одно или более значений, которые являются частью структуры данных. Инструкция завершается точкой с запятой «;». Даже при одинаковом синтаксисе инструкции отличаются от функций, инструкции не могут использоваться в выражениях. В выражениях могут использоваться только функции.	*instruction ()* *instruction (operand);* *instruction (operand1,operand2, operand3);*
конструкция (см. стр. 7-12)	Условный оператор, используется для включения программы структурированного текста (т.е. других операторов). Конструкция заканчивается точкой с запятой «;».	*IF...THEN* *CASE* *FOR...DO* *WHILE...DO* *REPEAT...UNTIL* *EXIT*
комментарий (см. стр. 7-28)	Текст, который поясняет часть программы на языке структурированного текста. • Используйте комментарий для того, чтобы облегчить понимание программы. • Комментарии не влияют на выполнение программы. • Комментарии могут быть в любом месте структурированного текста.	*//комментарий* *(*начало комментария . . . конец комментария*)* */*начало комментария . . . конец комментария*/*

Присваивание

Используйте присваивание для изменения значения, хранящегося в теге. Оператор присваивания имеет следующий синтаксис:

tag := expression;

где:

Элемент:	Описание:	
tag (тег)	представляет собой тег, который должен получить новое значение. Тег должен иметь тип BOOL, SINT, INT, DINT или REAL	
:=	символ присваивания	
expression (выражение)	представляет собой новое значение, присваиваемое тегу	
	Если *tag (тег)* имеет тип:	**Используйте выражение типа:**
	BOOL	Выражение типа BOOL
	SINT INT DINT REAL	числовое выражение
;	конец оператора присваивания	

Тег сохраняет присвоенное значение до тех пор, пока новое присваивание его не изменит.

Выражение может быть простым, например, непосредственным значением, или именем тега, а может быть сложным и включать несколько операторов и/или функций. Более подробно этот случай изложен в разделе «Выражения» на стр. 7-4.

Задание присваивания без сохранения

Присваивание без сохранения отличается от обычного присваивания, описанного выше, тем, что тег при присваивании без сохранения сбрасывается на ноль всякий раз, когда контроллер:

- входит в режим выполнения RUN
- выходит из шага SFC, если вы сконфигурировали SFC на автоматический сброс (Automatic reset) (Это применимо, только если вы вставили оператор присваивания в операции для данного шага или используете операцию для вызова процедуры структурированного текста через инструкцию JSR).

Оператор присваивания без сохранения имеет следующий синтаксис:

```
tag [:=] expression ;
```

где:

<table>
<tr><th>Элемент:</th><th colspan="2">Описание:</th></tr>
<tr><td>tag (тег)</td><td colspan="2">представляет собой тег, который должен получить новое значение. Тег должен иметь тип BOOL, SINT, INT, DINT или REAL</td></tr>
<tr><td>[:=]</td><td colspan="2">символ присваивания без сохранения</td></tr>
<tr><td rowspan="5">expression (выражение)</td><td colspan="2">представляет собой новое значение, присваиваемое тегу</td></tr>
<tr><td>Если tag (тег) имеет тип:</td><td>Используйте выражение типа:</td></tr>
<tr><td>BOOL</td><td>Выражение типа BOOL</td></tr>
<tr><td>SINT
INT
DINT
REAL</td><td>числовое выражение</td></tr>
<tr><td></td><td></td></tr>
<tr><td>;</td><td colspan="2">конец оператора присваивания</td></tr>
</table>

Присваивание символов ASCII строке

Используйте оператор присваивания для того, чтобы присвоить символ ASCII элементу члена DATA строкового тега. Чтобы присвоить символ, задайте значение символа или задайте имя тега, член DATA или элемент символа. Например:

Это правильно:	Это неправильно:
string1.DATA[0]:= 65;	*string1.DATA[0] := A;*
string1.DATA[0]:= string2.DATA[0];	*string1 := string2;*

Для того чтобы добавить или вставить строку символом в строковый тег, используйте следующие строковые инструкции ASCII:

Для того чтобы:	Используйте эту инструкцию:
добавить символы в конец строки	CONCAT
вставить символы в строку	INSERT

Выражения

Выражение является именем тега, уравнением или сравнением. Для того, чтобы записать выражение, используйте:

- имя тега, который хранит значение (переменную)
- число, которое вы хотите ввести (непосредственное значение)
- функции, такие как: ABS, TRUNC
- операторы, такие как: +, &, <, >, And, Or

Когда вы записываете выражение, следуйте этим общим правилам:

- Используйте любую комбинацию букв верхнего и нижнего регистров. Например, допустимы три варианта оператора "AND": AND, And и and.
- Для более сложных выражений используйте круглые скобки для создания групп внутри выражения. Это облегчит чтение и гарантирует, что выражение будет выполняться в нужной последовательности. См. раздел «Определение порядка выполнения» на стр. 7-10.

В структурированном тексте вы можете использовать выражения следующих типов:

Логическое выражение (BOOL): это выражение, имеющее на выходе булево значение 1 (истина) или 0 (ложь).

- В логическом выражении используются теги типа bool, операторы отношения и логические операторы для сравнения значений и проверки условий «истина» или «ложь». Например, tag1>65.
- Простые логические выражения могут содержать один тег типа BOOL.
- Обычно вы используете логические выражения для проверки условия выполнения другого алгоритма.

Числовое выражение: это выражение, использующее в расчете целые числа или числа с плавающей точкой.

- Числовое выражение использует арифметические операторы, арифметические функции и побитовые операторы. Например, tag1+5.
- Часто вы вкладываете арифметическое выражение в логическое. Например, `(tag1+5)>65`.

Используйте следующую таблицу для выбора операторов выражения:

Если вы хотите:	То:
Вычислить арифметическое значение	Обратитесь к разделу «Использование арифметических операторов и функций» на стр. 7-6.
Сравнить два значения или строки	Обратитесь к разделу «Использование операторов отношения» на стр. 7-7.
Проверить, какое значение имеет условие - «истина» или «ложь»	Обратитесь к разделу «Использование логических операторов» на стр. 7-9.
Сравнить биты внутри значения	Обратитесь к разделу «Использование поразрядных операторов» на стр. 7-10.

Использование арифметических операторов и функций

В арифметических выражениям вы можете комбинировать несколько операторов и функций.

Арифметические операторы рассчитывают новые значения.

Чтобы:	Используйте этот оператор:	Оптимальный тип данных:
сложить	+	DINT, REAL
вычесть/отрицать	-	DINT, REAL
умножить	*	DINT, REAL
возвести в степень (x в степень y)	**	DINT, REAL
разделить	/	DINT, REAL
разделить по модулю	MOD	DINT, REAL

Арифметические функции выполняют арифметические операции. Для этих функций необходимо задавать константу, тег не логического типа или выражение.

Для вычисления:	Используйте эту функцию:	Оптимальный тип данных:
абсолютного значения	ABS (numeric_expression)	DINT, REAL
арккосинуса	ACOS (numeric_expression)	REAL
арксинуса	ASIN (numeric_expression)	REAL
арктангенса	ATAN (numeric_expression)	REAL
косинуса	COS (numeric_expression)	REAL
перевода радиан в градусы	DEG (numeric_expression)	DINT, REAL
натурального логарифма	LN (numeric_expression)	REAL
десятичного логарифма	LOG (numeric_expression)	REAL
перевода градусов в радианы	RAD (numeric_expression)	DINT, REAL
синуса	SIN (numeric_expression)	REAL
квадратного корня	SQRT (numeric_expression)	DINT, REAL
тангенса	TAN (numeric_expression)	REAL
округления	TRUNC (numeric_expression)	DINT, REAL

Например:

Используйте этот формат:	Пример:	
	Для этой ситуации:	**Вам следует использовать запись**
value1 operator value2	Если gain_4 и gain_4_adj являются тегами типа DINT и ваше задание гласит: "Прибавить 15 к gain_4 и сохранить результат в gain_4_adj».	*gain_4_adj := gain_4+15;*
operator value1	Если alarm и high_alarm (сигналы тревоги) являются тегами типа DINT и ваше задание гласит: «Поменять знак у high_alarm и сохранить результат в alarm».	*alarm:= -high_alarm;*
function numeric_expression)	Если overtravel и overtravel_POS (перебег) являются тегами типа DINT и ваше задание гласит: «Рассчитать абсолютное значение перебега и сохранить результат в overtravel_POS».	*overtravel_POS := ABS(overtravel);*
value1 operator (function ((value2+value3)/2)	Если adjustment и position являются тегами типа DINT , sensor1 и sensor2 теги типа REAL и ваше задание гласит: "Найти абсолютное значение среднего значения sensor1 и sensor2, прибавить adjustment и сохранить результат в position».	*position := adjustment + ABS((sensor1 + sensor2)/2);*

Использование операторов отношения

Операторы отношения сравнивают два значения или две строки и выдают логический результат «истина» или «ложь». Результатом операции сравнения является булево значение:

Если сравнение:	Результат:
истина	1
ложь	0

Используйте следующие операторы отношения:

Для следующего сопоставления:	Используйте этот оператор:	Оптимальный тип данных:
равно	=	DINT, REAL, строка
меньше чем	<	DINT, REAL, строка
меньше или равно	<=	DINT, REAL, строка
больше чем	>	DINT, REAL, строка
больше или равно	>=	DINT, REAL, строка
не равно	<>	DINT, REAL, строка

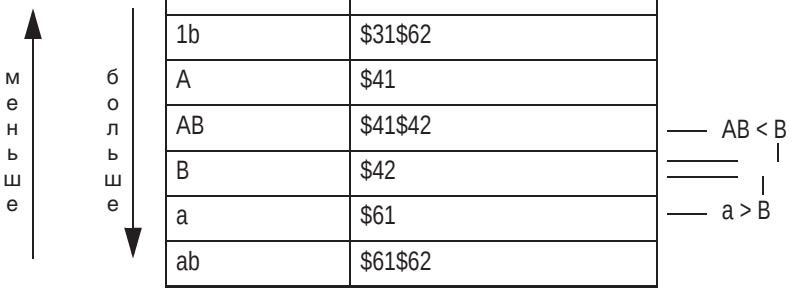
Например:

Используйте этот формат:	Пример:	
	Для этой ситуации:	Вам следует использовать запись
value1 operator value2	Если *temp* является тегом типа DINT и ваше задание гласит: "Если *temp* меньше 100°, то...».	`IF temp<100 THEN...`
stringtag1 operator stringtag2	Если *bar_code* и *dest* являются строковыми тегами и ваше задание гласит: «Если *bar_code* равен *dest*, то...».	`IF bar_code=dest THEN..`
char1 operator char2 Чтобы ввести символ ASCII прямо в выражение, введите десятичное значение этого символа.	Если *bar_code* строковый тег и ваше задание гласит: «Если *bar_code.DATA[0]* равен 'A', то ...».	`IF bar_code.DATA[0]=65 THEN...`
bool_tag := bool_expressions	Если *count* и *length* теги типа DINT, а *done* – это тег типа BOOL и ваше задание гласит: «Если *count* больше или равен *length*, то необходимо выполнить подсчет».	`done := (count >= length);`

Как производятся операции со строками

Если определяется, больше одна строка чем другая или меньше, то вычисляются шестнадцатиричные значения символов ASCII.

- Когда две строки сортируются в телефонном справочнике, порядок следования строк определяется тем, какая строка больше.



Символы ASCII	Шестнадцатиричные коды
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
ab	$61$62

- Строки равны, если их символы совпадают.
- Символы зависят от регистра. Прописная “A” ($41) *не* равна строчной “a” ($61).

Десятичные и шестнадцатиричные коды символов приведены в конце данного руководства.

Использование логических операторов

Логические операторы позволяют вам проверять, являются ли многочисленные условия истиной или ложью. Результатом логической операции является булево значение:

Если сравнение:	Результат:
истина	1
ложь	0

Используйте следующие логические операторы:

Для:	Используйте этот оператор:	Тип данных:
логической операции И	&, AND	BOOL
логической операции ИЛИ	OR	BOOL
логической операции исключающее ИЛИ	XOR	BOOL
логического дополнения	NOT	BOOL

Например:

Используйте этот формат:	Пример:	
	Для этой ситуации:	Вам следует использовать запись
BOOLtag	Если *photoeye* является тегом типа BOOL и ваше задание гласит: «Если *photoeye_1* установлен, то ...».	`IF photoeye THEN...`
NOT BOOLtag	Если *photoeye* является тегом типа BOOL и ваше задание гласит: «Если *photoeye_1* сброшен, то ...».	`IF NOT photoeye THEN...`
expression1 & expression2	Если *photoeye* является тегом типа BOOL , *temp* является тегом типа DINT и ваше задание гласит: «Если *photoeye* установлен и *temp* меньше 100° то...».	`IF photoeye & (temp<100) THEN...`
expression1 OR expression2	Если *photoeye* является тегом типа BOOL , *temp* является тегом типа DINT и ваше задание гласит: «Если *photoeye* установлен или *temp* меньше 100° то...».	`IF photoeye OR (temp<100) THEN...`
expression1 XOR expression2	Если *photoeye1* и *photoeye2* являются тегами типа BOOL и ваше задание гласит: «Если: • *photoeye1* установлен в то время как *photoeye2* сброшен или • *photoeye1* сброшен, в то время как *photoeye2* установлен то...».	`IF photoeye1 XOR photoeye2 THEN...`
BOOLtag := expression1 & expression2	Если *photoeye1*, *photoeye2* и open являются тегами типа BOOL и ваше задание гласит: «Если *photoeye1* и *photoeye2* оба установлены, присвоить *open* значение «истина».	`open := photoeye1 & photoeye2;`

Использование поразрядных операторов

Поразрядные операторы обрабатывают биты для двух значений.

Для:	Используйте этот оператор:	Оптимальный тип данных:
Поразрядного И	&, AND	DINT
Поразрядного ИЛИ	OR	DINT
Поразрядного исключающего ИЛИ	XOR	DINT
Поразрядного НЕ	NOT	DINT

Например:

Используйте этот формат:	Пример:	
	Для этой ситуации:	**Вам следует использовать запись**
величина1 оператор величина2 *value1 operator value2*	Если *input1*, *input2* и *result1* являются тегами типа DINT, и ваше задание гласит: «Рассчитать побитовый результат от *input1* и *input2*. Сохранить результат в *result1*».	`result1 := input1 AND input2;`

Определение порядка выполнения

Операции, которые вы записали в выражение, выполняются в предписанном порядке, и этот порядок не обязательно слева направо.

- Операции одного порядка выполняются слева направо.
- Если выражение содержит несколько операторов или функций, группируйте их в круглых скобках "( )". Это гарантирует правильный порядок выполнения и облегчит понимание выражения.

Порядок:	Операция:
1.	()
2.	функция(…)
3.	**
4.	-(смена знака)
5.	NOT
6.	*,/, MOD
7.	+,-(вычитание)
8.	<,<=,>,>=
9.	=,<>
10.	&, AND
11.	XOR
12.	OR

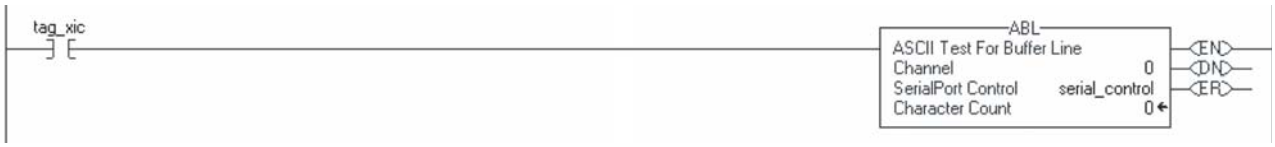
Инструкции

Операторами структурированного текста могут быть и инструкции. Список инструкций, имеющихся в структурированном тексте, приведен в начале данного руководства. Инструкции структурированного текста выполняются всякий раз, когда они сканируются. Инструкции структурированного текста внутри конструкции выполняются всякий раз, когда условия конструкции принимают значения «истина». Если условия конструкции имеют значение «ложь», операторы внутри конструкции не сканируются. Не существующее условие цепочки или перехода, которое запускало бы выполнение.

Это отличает инструкции структурированного текста от инструкций функционального блока, в котором для включения выполнения инструкции используется EnableIn. В структурированном тексте инструкции выполняются так, как будто бы EnableIn всегда установлен.

Это также отличает инструкции структурированного текста от инструкций релейной логики, в которой входное условие цепочки запускает выполнение. Некоторые инструкции релейной логики выполняются только в том случае, когда входное условие цепочки переключается со значения «ложь» на значение «истина». В релейной логике это так называемые переходные инструкции. В структурированном тексте инструкции будут выполняться при каждом своем сканировании, если вы не введете какое-либо предварительное условие на выполнение инструкции.

Например, инструкция ABL является переходной инструкцией в релейной логике. В этом примере инструкция ABL выполняется только при сканировании, когда *tag_xic* переходит из положения сброшен в положение установлен. Инструкция ABL не выполняется, если *tag_xic* находится в положении установлен или сброшен.



Для структурированного текста, если вы запишите этот пример следующим образом:

```
IF tag_xic THEN ABL(0,serial_control);
END_IF;
```

то инструкция ABL будет выполняться при каждом сканировании, когда *tag_xic* установлен, а не только тогда, когда *tag_xic* переходит из положения сброшен в положение установлен.

Если вы хотите, чтобы инструкция ABL выполнялась только тогда, когда *tag_xic* переходит из положения сброшен в положение установлен, вы должны ввести специальное условие. Используйте единичное включение инструкции.

```
osri_1.InputBit := tag_xic;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
ABL(0,serial_control);
END_IF;
```

Конструкции

Конструкции могут программироваться как одиночные или как вложенные в другие конструкции.

Если вы хотите:	Используйте эту структуру:	Имеющуюся в языках:	См. стр.
что-то сделать, если или когда имеет место заданное условие	IF...THEN	структурированный текст	7-13
выбрать что делать на основе числового значения	CASE...OF	структурированный текст	7-16
сделать что-либо заданное число раз, до того, как сделать что-нибудь еще	FOR...DO	структурированный текст	7-19
продолжать делать что-то, пока определенное условие имеет значение «истина»	WHILE...DO	структурированный текст	7-22
продолжать делать что-либо, пока определенное условие не примет значение «истина»	REPEAT...UNTIL	структурированный текст	7-25

IF...THEN

Используйте IF..THEN (если ... , то сделать ....) если или когда имеет место определенное условие.

Операнды:

Структурированный текст

```
IF bool_expression THEN
     <statement>;
END_IF;
```

Операнд:	Тип:	Формат:	Ввод:
bool_ expression	BOOL	тег выражение	тег типа BOOL или выражение, которое рассчитывает значение BOOL (логическое выражение)

Описание: Синтаксис:

```
IF bool_expression1 THEN
      <statement >;          <-- операторы для выполнения, когда bool_expression1 имеет значение "истина"
                .
                .
                .
optional ELSIF bool_expression2 THEN
      <statement>;           <-- операторы для выполнения, когда bool_expression2 имеет значение "истина"
                .
                .
                .
optional ELSE
      <statement>;           <-- операторы для выполнения, когда оба выражения имеют значение "ложь"
                .
                .
                .
END_IF;
```

При использовании ELSIF или ELSE следуйте следующим указаниям:

1. Чтобы произвести выбор из нескольких групп операторов, добавьте один или более операторов ELSIF.
   - Каждый оператор ELSIF представляет альтернативный путь.
   - Задавайте столько путей ELSIF, сколько вам нужно.
   - Контроллер выполняет первое значение «истина» для IF или ELSIF и игнорирует оставшиеся. ELSIF и ELSE.
2. Для того, чтобы что-либо сделать, когда все условия для IF или ELSIF имеют значения «ложь», добавьте оператор ELSE.

В следующей ниже таблице представлены различные комбинации IF, THEN, ELSIF и ELSE.

Если вы хотите:	И:	Используйте эту конструкцию:
что-либо сделать, если или когда заданные условия имеют значение «истина»	ничего не делать, если эти условия имеют значение «ложь»	IF…THEN
	что-то сделать другое, если эти условия имеют значение «ложь»	IF…THEN…ESLE
сделать выбор из альтернативных операторов (или групп операторов) на основе входных условий	ничего не делать, если эти условия имеют значение «ложь»	IF…THEN…ELSIF
	присвоить операторы по умолчанию, если все эти условия имеют значение «ложь»	IF…THEN…ELSIF…ELSE

Арифметические флаги состояния: не присваиваются.

Условия ошибки: нет

Пример 1: IF…THEN

Если вы хотите:	Используйте:

Если количество бракованных деталей больше 3, то:
конвейер остановить (conveyor = off (0))
включить сигнал тревоги (alarm = on (1))
	`IF rejects > 3 THEN`
	`  conveyor := 0;`
	`  alarm := 1;`
	`END_IF;`

Пример 2: IF…THEN…ELSE

Если вы хотите:	Используйте:

Если контакт направления движения конвейера имеет значение forward (1), то:
выключить лампочку (light = off)
в противном случае лампочку включить (light = on)
	`IF conveyor_direction THEN`
	`  light := 0;`
	`ELSE`
	`  light [:=] 1;`
	`END_IF;`

Символ [:=] приказывает контроллеру сбрасывать *light* всякий раз, когда контроллер

- входит в режим выполнения RUN,
- выходит из шага ПФС, если вы сконфигурировали ПФС на *Automatic reset* (автоматический сброс). (Применимо только если вы вставляете оператор в операцию (action) или используете операцию для вызова процедуры структурированного текста посредством инструкции ПФС.)

Пример 3: IF…THEN…ELSIF

Если вы хотите:	Используйте:

Если нижний концевой выключатель уровня сахара имеет значение low (on) и верхний концевой выключатель уровня сахара имеет значение not high (on), то открыть впускной клапан (open (on))
Пока верхний концевой выключатель уровня сахара имеет значение high (off).
	`IF Sugar.Low & Sugar.High THEN`
	`Sugar.Inlet [:=] 1;`
	`ELSIF NOT(Sugar.High) THEN`
	`Sugar.Inlet := 0;`
	`END_IF;`

Символ [:=] приказывает контроллеру сбрасывать *Sugar.Inlet* всякий раз, когда контроллер

- входит в режим выполнения RUN,
- выходит из шага ПФС, если вы сконфигурировали ПФС на *Automatic reset* (автоматический сброс). (Применимо только если вы вставляете оператор в операцию (action) или используете операцию для вызова процедуры структурированного текста посредством инструкции ПФС.)

Пример 4: IF…THEN…ELSIF…ELSE

Если вы хотите чтобы:	Используйте:

Если температура бака больше 100, то параметру работы насоса присвоить значение медленно (slow),
Если температура бака больше 200, то параметру работы насоса присвоить значение быстро (fast),
в противном случае насос отключить (off).
	`IF tank.temp > 200 THEN`
	`pump.fast :=1; pump.slow :=0; pump.off :=0;`
	`ELSIF tank.temp > 100 THEN`
	`pump.fast :=0; pump.slow :=1; pump.off :=0;`
	`ELSE`
	`pump.fast :=0; pump.slow :=0; pump.off :=1;`
	`END_IF;`

CASE...OF

Используйте конструкцию CASE для выбора последующих действий на основе числового значения.

Операнды:

Структурированный текст

```
CASE numeric_expression OF
    selector1: statement;
    selectorN: statement;
ELSE
    statement;
END CASE;
```

Операнд:	Тип:	Формат:	Ввод:
numeric_expression	SINT INT DINT REAL	тег выражение	тег или выражение, рассчитывающее числовое значение
selector	SINT INT DINT REAL	непосредственный	тот же самый тип, что и *numeric_expression*

ВАЖНО Если вы используете величины типа REAL, используйте для операнда selector некий диапазон значений, поскольку для значений типа REAL более верно говорить о попадании в диапазон, а не о строгом совпадении одной величины с другой.

Описание: Синтаксис:

```
CASE numeric_expression OF
    selector1 : <statement>;   <-- операторы для выполнения, если numeric_expression = selector1
                .
                .
                .
    selector2 : <statement>;   <-- операторы для выполнения, если numeric_expression = selector2
                .
                .
                .
    selector3 : <statement>;   <-- операторы для выполнения, если numeric_expression = selector3
                .
                .
                .
ELSE                           (необязательно)
                <statement>;   <-- операторы для выполнения, если numeric_expression неравен любому из selector
                .
                .
                .
END_CASE;
```

задавайте столько альтернативных значений операнда selector, сколько вам нужно

Допустимые значения операнда selector представлены в таблице на следующей странице.

Синтаксис для ввода значений операнда selector:

Когда selector:	**Вводите:**
одно значение	`value: statement`
несколько различных значений	`value1, value2, valueN : <statement>` Используйте запятую (,) для разделения значений.
диапазон значений	*value1..valueN* : *<statement>* Используйте две точки (..) для определения диапазона.
различные значения + диапазон значений	*valuea, valueb, value1..valueN* : *<statement>*

Конструкция CASE подобна оператору switch в языках программирования C или C++. Однако, при использовании CASE контроллер выполняет *только те* операторы, которые соответствуют *первому совпадению* с значением selector. *После выполнения операторов этого операнда выполнение всегда останавливается* и управление передается на оператор END_CASE.

Арифметические флаги состояния: не присваиваются.

Условия ошибки: нет

Пример:

Если вы хотите:	Введите:
If recipe number = 1 then	`CASE recipe_number OF`
Ingredient A outlet 1 = open (1)	`1: Ingredient_A.Outlet_1 :=1;`
Ingredient B outlet 4 = open (1)	`Ingredient_B.Outlet_4 :=1;`
If recipe number = 2 or 3 then	`2,3: Ingredient_A.Outlet_4 :=1;`
Ingredient A outlet 4 = open (1)	`Ingredient_B.Outlet_2 :=1;`
Ingredient B outlet 2 = open (1)	
If recipe number = 4, 5, 6, or 7 then	`4..7: Ingredient_A.Outlet_4 :=1;`
Ingredient A outlet 4 = open (1)	`Ingredient_B.Outlet_2 :=1;`
Ingredient B outlet 2 = open (1)	
If recipe number = 8, 11, 12, or 13 then	`8,11..13 Ingredient_A.Outlet_1 :=1;`
Ingredient A outlet 1 = open (1)	`Ingredient_B.Outlet_4 :=1;`
Ingredient B outlet 4 = open (1)	
Otherwise all outlets = closed (0)	`ELSE`
	`Ingredient_A.Outlet_1 [:=]0;`
	`Ingredient_A.Outlet_4 [:=]0;`
	`Ingredient_B.Outlet_2 [:=]0;`
	`Ingredient_B.Outlet_4 [:=]0;`
	`END_CASE;`

Символ [:=] приказывает контроллеру сбрасывать light всякий раз, когда контроллер

- входит в режим выполнения RUN,
- выходит из шага ПФС, если вы сконфигурировали ПФС на *Automatic reset* (автоматический сброс). (Применимо только если вы вставляете оператор в операции (action) или используете операцию для вызова процедуры структурированного текста посредством инструкции ПФС.)

FOR...DO

Используйте цикл FOR...DO для выполнения каких-либо операций заданное число раз перед тем, как перейти к другим операциям.

Операнды:

Структурированный текст

```
FOR count:= initial_value TO
final_value BY increment DO
    <statement>;
END_FOR;
```

Операнд:	Тип:	Формат:	Ввод:
count	SINT INT DINT	тег	тег для хранения позиции счетчика (count) при выполнении FOR...DO
initial_ value	SINT INT DINT	тег выражение непосредственный	рассчитывает первое значение для count
final_ value	SINT INT DINT	тег выражение непосредственный	рассчитывает последнее значение для count, определяющее, когда выходить из цикла
increment	SINT INT DINT	тег выражение непосредственный	(*необязательно*) приращение count Если вы не задаете increment, то приращение равно 1.

.

ВАЖНО

Проверяйте, не делаете ли вы слишком много итераций в пределах одного сканирования.

- Контроллер не будет выполнять никаких других операторов в программе, пока не завершит цикл.
- Если время, требующееся для завершения цикла, больше, чем время контролирующего таймера для этого задания, будет иметь место основная ошибка.
- Тщательно обдумывайте использование таких структур как IF...THEN.

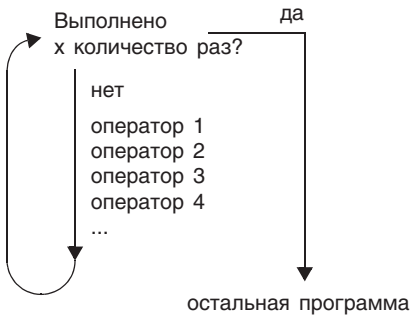
Описание: Синтаксис:

```
FOR count := initial_value
   TO final_value
   BY increment                 (необязательно)
   DO
   <statement>;
   IF bool_expression THEN      (необязательно)
       EXIT;
   END_IF;
END_FOR;
```

Если вы не задаете приращение, значение увеличивается на 1.

Если существуют условия, при который вы хотите выйти из цикла раньше, используйте такую структуру как IF...THEN для задания условий перехода к оператору EXIT.

Следующая ниже схема демонстрирует выполнение цикла FOR...DO и объяснение, как правильно использовать оператор EXIT для более раннего выхода из цикла.



Цикл FOR…DO выполняется заданное число раз.

Для остановки цикла до того, как счетчик count достигнет последнего значения, используйте оператор EXIT.

Арифметические флаги состояния: не присваиваются.

Условия ошибки:

Основная ошибка имеет место если:	Тип ошибки	Код ошибки
Конструкция цикла слишком длинная	6	1

Пример 1:

Если вы хотите:	Введите:

Очистить биты 0 - 31 в массиве BOOL:
1. *Присвоить* тегу *индекса* (subscript) значение 0.
2. Очистить *элемент массива* [индекс] (array[subscript]). Например, когда *индекс* = 5, очистить элемент массива [5].
3. Прибавить 1 к *значению индекса*.
4. Если индекс J 31, повторять 2 и 3.
В противном случае, остановиться.
	`For subscript:=0 to 31 by 1 do`
	`array[subscript] := 0;`
	`End_for;`

Пример 2:

Если вы хотите:	Введите:

Структура хранения данных пользователя содержит следующую информацию:
• Идентификационный штриховой код продукта (строковый тип)
• Количество данного продукта на складе (тип данных DINT)
Массив (Inventory), описанной выше структуры содержит один элемент для каждого продукта. Вы хотите найти заданный продукт (используя штриховой код) и определить его количество на складе.
1. Определить размер массива *Inventory* (ассортимент продуктов) и сохранить результат в *Inventory_Items* (тег типа DINT).
2. Присвоить индексу 0.
3. Если штриховой код *Barcode* совпадает с идентификатором продукта ID в массиве, то:
a. Присвоить тегу *Quantity* (количество) = *Inventory*[position].*Qty*. Это количество продукта на складе.
b. Стоп.
Barcode является строковым тегом, в котором хранится штриховой код продукта, который вы ищете. Например, если *position* = 5, сравнивается *Barcode* с *Inventory[5].ID*.
4. Прибавить 1 к *position*.
5. Если *position* J (*Inventory_Items - 1*), повторить 3 и 4. Поскольку нумерация начинается с 0, последний элемент на 1 меньше чем номер в массиве. В противном случае, стоп.
	`SIZE(Inventory,0,Inventory_Items);`
	`For position:=0 to Inventory_Items - 1 do`
	`If Barcode = Inventory[position].ID then`
	`Quantity := Inventory[position].Qty;`
	`Exit;`
	`End_if;`
	`End_for;`

WHILE...DO

Используйте цикл WHILE...DO для того, чтобы продолжать выполнять какие-либо операции, пока заданные условия сохраняют значение «истина».

Операнды:

```
WHILE bool_expression DO
    <statement>;
END_WHILE;
```

Структурированный текст

Операнд:	Тип:	Формат:	Ввод:
bool_expression	BOOL	тег выражение	тег типа BOOL или выражение, которое рассчитывает значение типа BOOL

.

ВАЖНО

Проверяйте, не делаете ли вы слишком много итераций в пределах одного сканирования.

- Контроллер не будет выполнять никаких других операторов в программе, пока не завершит цикл.
- Если время, требующееся для завершения цикла, больше чем время контролирующего таймера для этого задания, будет иметь место основная ошибка.
- Тщательно обдумывайте использование таких структур как IF...THEN.

Описание: Синтаксис:

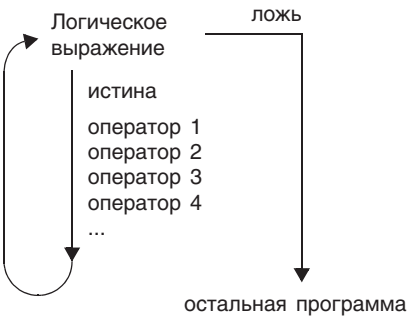
```
WHILE bool_expression1 DO
   <statement>;

   IF bool_expression2 THEN
     EXIT;
   END_IF;

END_WHILE;
```

Операторы для выполнения, пока *bool_expression1* имеет значение "истина".

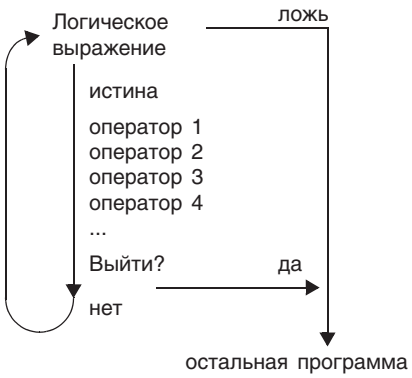
Необязательно

Если существуют условия, при который вы хотите выйти из цикла раньше, используйте такую структуру как IF...THEN для задания условий перехода к оператору EXIT.

Следующая ниже схема демонстрирует выполнение цикла WHILE...DO и использование оператора EXIT для более раннего выхода из цикла.



Пока логическое выражение *bool_expression* сохраняет значение "истина", контроллер выполняет только операторы внутри цикла WHILE...DO.



Для остановки цикла, когда сохраняется значение "истина", используйте оператор EXIT.

Арифметические флаги состояния: не присваиваются.

Условия ошибки:

Основная ошибка имеет место если:	Тип ошибки	Код ошибки
Конструкция цикла слишком длинная	6	1

Пример 1:

Если вы хотите чтобы:	Введите:

Цикл WHILE...DO, в первую очередь, рассчитал условия выполнения. Если эти условия имеют значение «истина», контроллер выполнит операторы внутри цикла.
Это отличается от цикла REPEAT...UNTIL, поскольку цикл REPEAT...UNTIL выполняет операторы в конструкции, а потом определяет, имеют ли условия значение «истина» перед следующим выполнением операторов. Операторы в цикле REPEAT...UNTIL всегда выполняются хотя бы один раз. Операторы в цикле WHILE...DO могут не выполняться ни разу.
	`pos := 0;`
	`While ((pos <= 100) & structarray[pos].value <> targetvalue)) do`
	`pos := pos + 2;`
	`String_tag.DATA[pos] := SINT_array[pos];`
	`end_while;`

Пример 2:

Если вы хотите:	Введите:

Переместить символы ASCII из массива типа SINT в строковый тег. (В массиве SINT каждый элемент содержит один символ.) Остановиться по достижении символа возврата каретки.
1. Значению *Element_number* (номер элемента) присваивается 0.
2. Подсчитывается количество элементов в массиве *SINT_array* (массив, содержащий символы ASCII) и результат сохраняется в *SINT_array_size* (тег типа DINT).
3. Если символ в *SINT_array[element_number]* = 13 (десятичный код возврата каретки), то остановиться.
4. Присвоить *String_tag[element_number]* = символ в *SINT_array[element_number]*.
5. Прибавить 1 к *element_number*. Это позволит контроллеру проверить следующий символ в *SINT_array*.
6. Присвоить элементу Length тега *String_tag* значение *element_number.* (Это запись количества символов в *String_tag*.)
7. Если *element_number* = *SINT_array_size*, остановиться. (Вы в конце массива и он не содержит символа возврата каретки.)
8. Перейти к 3.
	`element_number := 0;`
	`SIZE(SINT_array, 0, SINT_array_size);`
	`While SINT_array[element_number] <> 13 do`
	`  String_tag.DATA[element_number] :=`
	`  SINT_array[element_number];`
	`  element_number := element_number + 1;`
	`  String_tag.LEN := element_number;`
	`  If element_number = SINT_array_size then`
	`  exit;`
	`  end_if;`
	`end_while;`

REPEAT...UNTIL

Используйте цикл REPEAT...UNTIL для того, чтобы продолжать выполнять какие-либо операции, пока заданные условия сохраняют значение «истина».

Операнды:

```
REPEAT
    <statement>;
UNTIL bool_expression
END_REPEAT;
```

Структурированный текст

Операнд:	Тип:	Формат:	Ввод:
bool_expression	BOOL	тег выражение	тег типа BOOL или выражение, которое рассчитывает значение типа BOOL (логическое выражение)

.

ВАЖНО

Проверяйте, что вы *не делаете* слишком много итераций в пределах одного сканирования.

- Контроллер *не будет* выполнять никаких других операторов в программе, пока не завершит цикл.
- Если время, требующееся для завершения цикла, больше чем время контролирующего таймера для этого задания, будет иметь место основная ошибка.
- Тщательно обдумывайте использование таких структур как IF...THEN.

Описание: Синтаксис:

```
REPEAT
   <statement>;
   IF bool_expression2 THEN
     EXIT;
   END_IF;
UNTIL bool_expression1
END_REPEAT;
```

Необязательно

Операторы для выполнения, пока *bool_expression1* имеет значение "ложь".

Если существуют условия, при который вы хотите выйти из цикла раньше, используйте такую структуру как IF...THEN для задания условий перехода к оператору EXIT.

Следующая ниже схема демонстрирует выполнение цикла REPEAT...UNTIL и то, как использовать оператор EXIT для более раннего выхода из цикла.

Пока логическое выражение *bool_expression* сохраняет значение "ложь", контроллер выполняет только операторы внутри цикла REPEAT...UNTIL .

Для остановки цикла, когда сохраняется значение "истина", используйте оператор EXIT.

Арифметические флаги состояния: не присваиваются.

Условия ошибки:

Основная ошибка имеет место если:	Тип ошибки	Код ошибки
Конструкция цикла слишком длинная	6	1

Пример 1:

Если вы хотите чтобы:	Введите:

Цикл REPEAT...UNTIL выполняет операторы структуры, а затем, перед выполнением этих операторов еще раз, проверяет, имеют ли условия значение «истина».
Это отличается от цикла WHILE...DO, потому что WHILE...DO сначала рассчитывает условия. Если эти условия имеют значение «истина» контроллер выполняет операторы внутри цикла. Операторы в цикле REPEAT...UNTIL всегда выполняются хотя бы один раз. Операторы в цикле WHILE...DO могут не выполняться ни разу.
	`pos := -1;`
	`REPEAT`
	`pos := pos + 2;`
	`UNTIL ((pos = 101) OR`
	`(structarray[pos].value = targetvalue))`
	`end_repeat;`

Пример 2:

Если вы хотите:	Введите:

Переместить символы ASCII из массива типа SINT в строковый тег. (В массиве SINT каждый элемент содержит один символ.) Остановиться по достижении символа возврата каретки.
1. Значению *Element_number* (номер элемента) присваивается 0.
2. Подсчитывается количество элементов в массиве *SINT_array* (массив, содержащий символы ASCII) и результат сохраняется в *SINT_array_size* (тег типа DINT).
3. Присвоить *String_tag[element_number]* = символ в *SINT_array[element_number].*
4. Прибавить 1 к *element_number*. Это позволит контроллеру проверить следующий символ в *SINT_array*.
5. Присвоить элементу *Length тега String_tag* значение *element_number*. (Это запись количества символов в *String_tag*.)
6. Если *element_number = SINT_array_size*, остановиться. (Вы в конце массива и он не содержит символа возврата каретки.)
7. Если символ в *SINT_array[element_number]* = 13 (десятичный код возврата каретки), то остановиться. В противном случае, перейти к 3.
	`element_number := 0;`
	`SIZE(SINT_array, 0, SINT_array_size);`
	`Repeat`
	`String_tag.DATA[element_number] :=`
	`SINT_array[element_number];`
	`element_number := element_number + 1;`
	`String_tag.LEN := element_number;`
	`If element_number = SINT_array_size thenexit;`
	`end_if;`
	`Until SINT_array[element_number] = 13`
	`end_repeat;`

Комментарии

Используйте комментарии для того, чтобы сделать программу на языке структурированного текста более понятной при чтении.

- Комментарии позволяют вам использовать простой язык для описания работы программы на языке структурированного текста.
- Комментарии не влияют на выполнение программы.

Чтобы добавить комментарии к программе на языке структурированного текста:

Чтобы добавить комментарий:	Используйте следующие форматы:
на одной строке	`//comment`
в конце строки структурированного текста	`(*comment*)` `/*comment*/`
внутри одной строки структурированного текста	`(*comment*)` `/*comment*/`
если занимает более одной строки	`(*начало комментария . . .конец комментария*)` `/* начало комментария . . . конец комментария*/`

Например:

Формат:	Пример:

`//comment`	**В начале строки**
	`//Check conveyor belt direction`
	`IF conveyor_direction THEN...`
	
	В конце строки
	`ELSE //If conveyor isn't moving, set alarm light`
	`light := 1;`
	`END_IF;`

`(*comment*)`	`Sugar.Inlet[:=]1;(*open the inlet*)`
	
	`IF Sugar.Low (*low level LS*)& Sugar.High (*high level LS*)THEN...`
	
	`(*Controls the speed of the recirculation pump. The speed depends on the temperature in the tank.*)`
	`IF tank.temp > 200 THEN...`

`/*comment*/`	`Sugar.Inlet:=0;/*close the inlet*/`
	
	`IF bar_code=65 /*A*/ THEN...`
	
	`/*Gets the number of elements in the Inventory array and stores the value in the Inventory_Items tag*/`
	`SIZE(Inventory,0,Inventory_Items);`

Программирование на языке релейной логики

Когда используется данная процедура

Используйте данную процедуру для:

- создания алгоритма процедуры на языке релейной логики
- ввода алгоритма в подпрограмму.

Перед использованием данной процедуры

Перед началом использования данной процедуры убедитесь, что у вас есть разрешение на выполнение следующих задач:

☐ Navigate the Controller Organizer (Осуществлять навигацию контроллера)

☐ Identify the Programming Languages That Are Installed (Идентифицировать установленные языки программирования)

За более подробной информацией относительно данных задач обратитесь к разделу «Начало» (Getting Started) на стр. 1-1.

Как пользоваться данной процедурой

Для программирования на языке релейной логики:

Определения

Перед написанием или вводом программы на языке релейной логики еще раз поясним следующие термины:

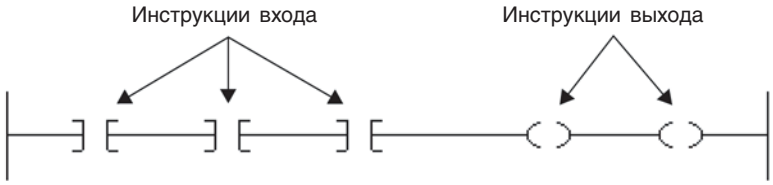
- Инструкция
- Ветвь
- Условие цепочки

Инструкция

Вы организуете алгоритм на языке релейной логики в виде многоступенчатой схемы цепочек, размещая инструкции в пределах каждой цепочки. Существует два основных типа инструкций:

Инструкция входа: Инструкция, которая проверяет, сравнивает или проверяет конкретные условия для вашей машины или процесса.

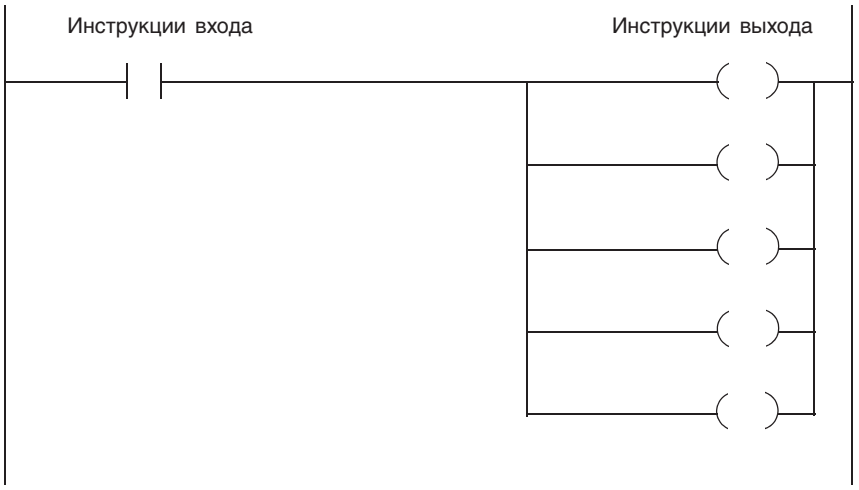
Инструкция выхода: Инструкция, которая реализует операции, такие как включение устройства, выключение устройства, копирование данных или расчет значения.



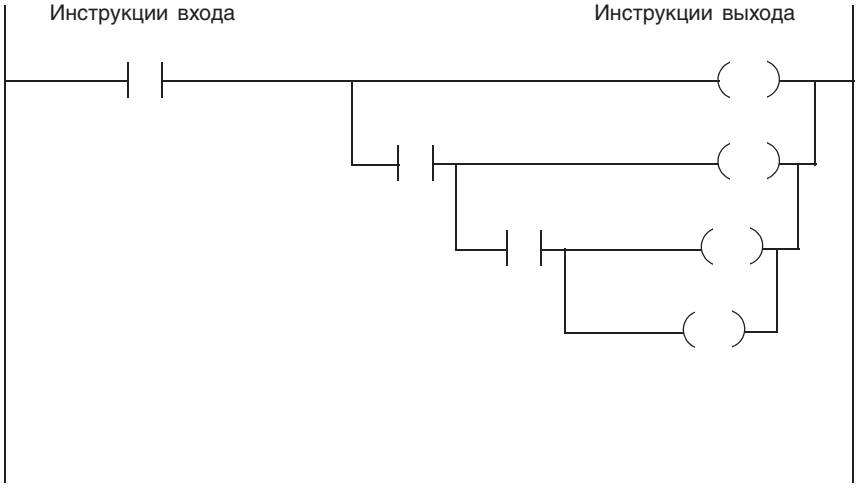
Ветвь

Ветвь представляет собой две или более параллельных инструкции.

Не существует ограничений на количество уровней параллельных ветвей, которые вы можете ввести. На следующем ниже рисунке показана параллельная ветвь с пятью уровнями. Основной цепочкой является ветвь первого уровня, за которой следуют четыре.

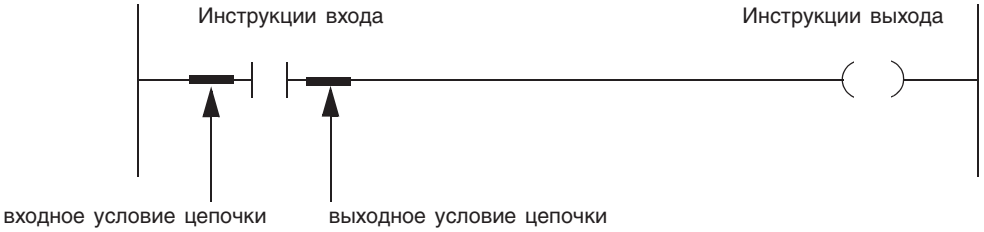


Вы можете вкладывать друг в друга ветви до 6 уровней. На рисунке, следующем ниже, показаны такие вложенные ветви. Самая нижняя инструкция вывода находится на вложенной ветви третьего уровня.



Условие цепочки

Контроллер оценивает инструкции релейной логики на основе условия цепочки, предшествующего инструкции (входное условие цепочки).



На входное условие соответствующей инструкции в цепочке оказывает влияние только инструкция входа:

- Если входное условие цепочки для инструкции входа имеет значение «истина» то контроллер вычисляет значение инструкции и устанавливает выходное условие цепочки совпадающим с результатом этого вычисления.
  - Если результатом вычисления является значение «истина», выходное условие цепочки имеет значение «истина».
  - Если результатом вычисления является значение «ложь», выходное условие цепочки имеет значение «ложь».
- Инструкция выхода не изменяет выходное значение цепочки.
  - Если входное условие цепочки для инструкции выхода «истина», то выходное условие цепочки принимает значение «истина».
  - Если входное условие цепочки для инструкции выхода «ложь», то выходное условие цепочки принимает значение «ложь».

Написание алгоритма на языке релейной логики

Для создания программы на языке релейной логики выполните следующие действия:

☐ Выберите требующиеся инструкции

☐ Скомпонуйте инструкции входа

☐ скомпонуйте инструкции выхода

☐ Выберите имя тега для операнда.

Выбор требующихся инструкций

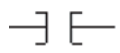
1. Отделите условия для проверки от действий.

2. Выберите соответствующие инструкции входа для каждого условия соответствующие инструкции выхода для каждой операции.

Конкретные инструкции описаны в руководствах:

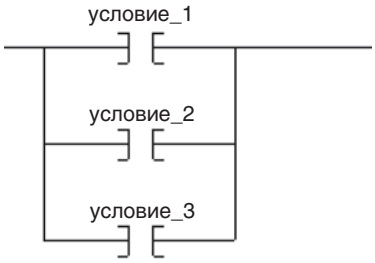
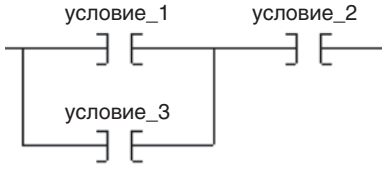
- *Общие инструкции контроллеров Logix5000, Справочное руководство, документ* 1756-RM003
- Инструкции обработки и управления для контоллеров *Logix5000, Справочное руководство, документ* 1756-RM006
- Комплект инструкций по управлению движением для контроллеров *Logix5000, Справочное руководство, документ* 1756-RM007.

Для того, чтобы помочь вам научиться писать программы а языке релейной логики, в примерах, используемых в данной главе, используются две простые инструкции.

Символ:	Имя:	Мнемоника:	Описание:	
	Examine If Closed	XIC	Инструкция входа, которая просматривает один бит данных	
			Если этот бит:	**То инструкция (выходное условие цепочки):**
			on (1)	истина
			off (0)	ложь
	Output Energize	OTE	Инструкция выхода, которая просматривает один бит данных	
			Если инструкция слева (входное условие цепочки)	**о эта инструкция устанавливает бит:**
			истина	on (1)
			ложь	off (0)

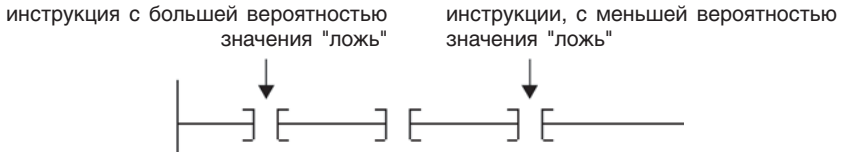
Компоновка инструкций входа

Компонуйте инструкции входа на цепочке используя следующую схему:

Для проверки нескольких условий когда:	Скомпонуйте инструкции входа:
Условия должны выполняться по порядку Например, If condition_1 AND condition_2 AND condition_3...	в виде последовательности 
Должно выполняться любое из нескольких условий Например, If condition_1 OR condition_2 OR condition_3...	параллельно 
Комбинация, описанных выше Например If condition_1 AND condition_2... OR If condition_3 AND condition_2...	комбинация 

СОВЕТ

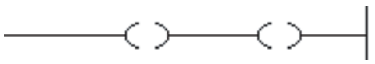
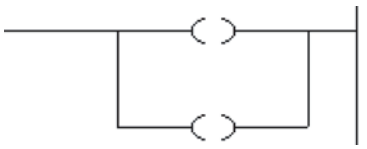
Контроллер выполняет все инструкции цепочки независимо от их входных условий цепочки. Для оптимального выполнения ряда инструкций последовательность должна быть такой, чтобы инструкция с большей вероятностью значения «ложь» располагалась слева от инструкции, с меньшей вероятностью значения «ложь».



Если контроллер находит инструкцию со значением «ложь» он выполняет оставшиеся инструкции в ряду с их входными условиями «ложь». Обычно быстрее выполняются инструкции с входными условиями цепочки «ложь», а не «истина».

Компоновка инструкций выхода

Справа от инструкции входа необходимо разместить хотя бы одну инструкцию выхода. В релейной логике вы можете размещать несколько инструкций выхода в цепочке:

Опция:	Пример:
Размещение инструкций выхода последовательно в цепочке	
Размещение инструкций выхода в ветвях (параллельно)	
Размещение инструкций выхода между инструкциями входа, когда последняя инструкция цепочки является инструкцией выхода.	

Выбор имени тега для операнда

Большинство инструкций требует наличие одного или более операндов следующих типов:

- Имя тега (переменная)
- Непосредственное значение (постоянная)
- Имя процедуры, метки и т.д.

В следующей ниже таблице представлены форматы для имени тега:

Для:	Задайте:
Тега	*tag_name*
Битовый номер наибольшего типа данных	*tag_name.bit_number*
Элемента конструкции	*tag_name.member_name*
Элемента одномерного массива	*tag_name[x]*
Элемента двухмерного массива	*tag_name[x,y]*
Элемента трехмерного массива	*tag_name[x,y,z]*
Элемента массива в пределах конструкции	*tag_name.member_name[x]*
Члена элемента массива	*tag_name[x,y,z].member_name*

Где:

x позиция элемента для первой размерности,

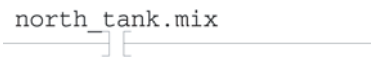
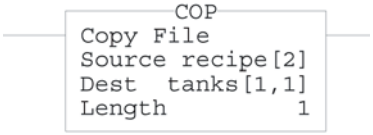
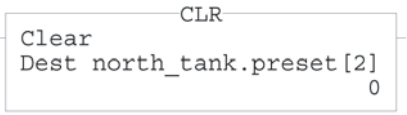
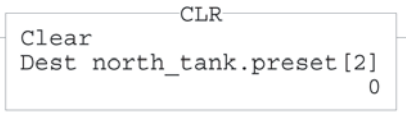
y позиция элемента для второй размерности,

z позиция элемента для третьей размерности.

Для конструкции в конструкции дополнительно добавьте *.member_name* .

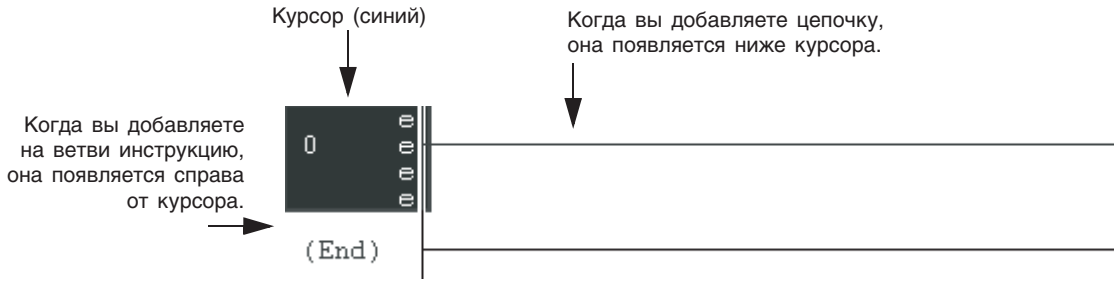
.

ПРИМЕР Смена имени тега для операнда

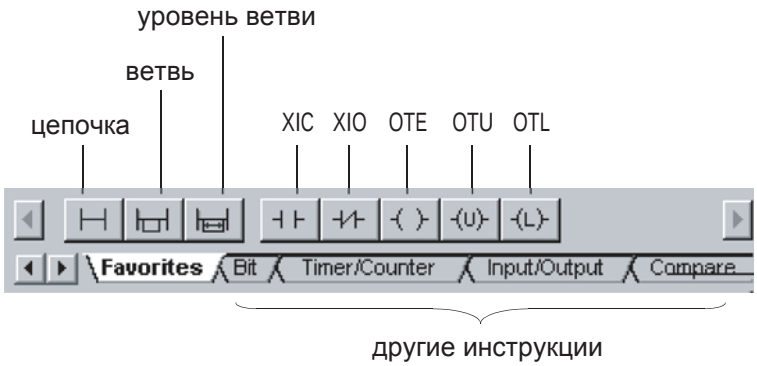
Чтобы получить доступ к:	**Имя тега выглядит следующим образом:**
Тегу *machine_on*	
Первому биту тега *one_shots*	
Элементу *DN (биту)* таймера *running_seconds*	
Элементу *mix* тега *north_tank*	
Элементу 2 массива *recipe* и элементу 1,1 в массиве *tanks*	
Элементу 2 массива *preset* в теге *north_tank*	
Элементу *part_advance* массива *drill*	

Ввод релейной логики

Новая процедура содержит цепочку, подготовленную для ввода инструкций.



Для ввода элементов процедуру релейной логики используйте панель инструментов Language Element (Элементы языка).



Чтобы добавить элемент:

☐ Присвойте элемент курсору

☐ Отбуксируйте элемент.

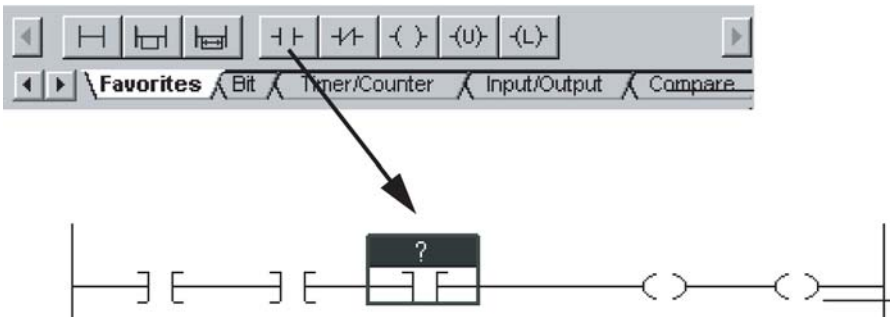
Присвоение элемента курсору

1. Щелкните (выберите) инструкцию, ветвь или цепочку, расположенную выше или слева от того места, где вы хотите добавить элемент.
2. На панели инструментов Language Element (Элементы языка) щелкните на кнопке того элемента, который вы хотите добавить.

Буксировка элемента

Отбуксируйте элемент в точку его расположения. Зеленая точка указывает подходящее местоположение (точка сброса).

Например



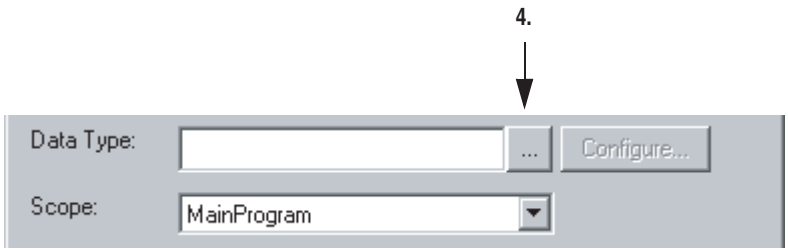
Присваивание операндов

Для присваивания операндов у вас имеются следующие опции:

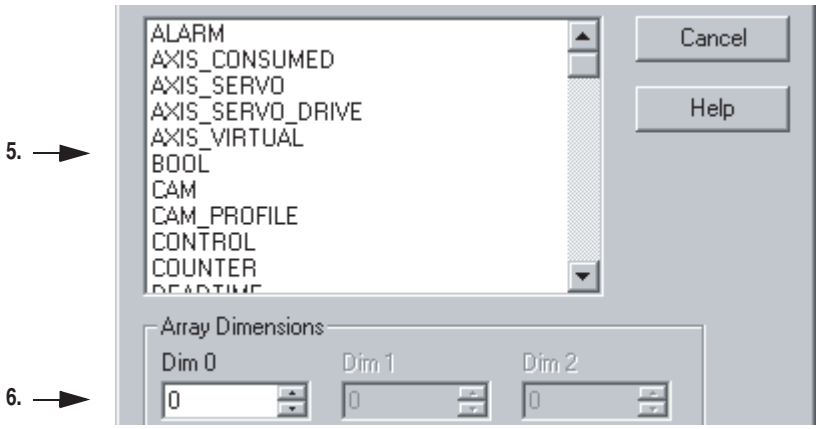
- ☐ Create and Assign a New Tag (Создать и присвоить новый тег)
- ☐ Choose a Name or an Existing Tag (Выбрать имя или существующий тег)
- ☐ Drag a Tag From the Tags Window (Перенести тег из окна тегов)
- ☐ Assign an Immediate (Constant) Value (Присвоить непосредственное значение (постоянную))

Создание и присвоение нового тега

1. Щелкните в области операнда данной инструкции.
2. Введите имя тега и нажмите [Enter].
3. Щелкните правой клавишей мыши на имени тега и выберите *New "tag_name" (Новое имя тега)*.



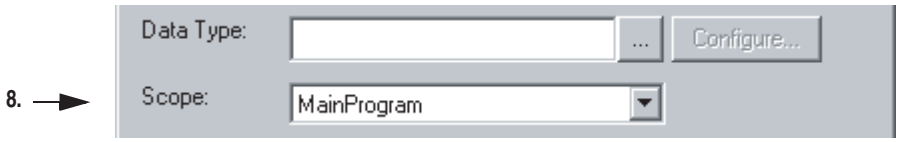
4. Щелкните на кнопке ... .



5. Выберите для этого тега тип данных.

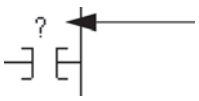
6. Если вы хотите определить тег как массив, введите число элементов для каждой размерности.

7. Выберите OK ОК.



8. Выберите для тега область применимости.

9. Выберите OK .



Выбрать имя или Существующий тег

1. Дважды щелкните в области операнда.

   Откроется окно ввода текста

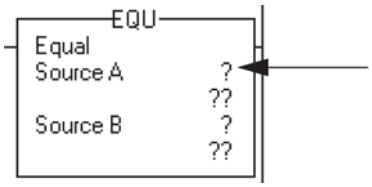
2. Щелкните на ▼
3. Выберите имя:

Чтобы выбрать:	Сделайте:
Метку, имя процедуры или тот же тип имени	Щелкните на имени
тег	Дважды щелкните на имени тега
Номер бита	А. Щелкните на имени тега. Б. Справа от имени тега щелкните на ▼ В. Щелкните на нужном бите.

4. Нажмите [Enter] или щелкните в другом месте схемы.

Буксировка тега из окна тегов

1. Найдите данный тег в окне Tags (Теги).
2. Щелкните на этом теге два или три раза, пока не сработает выделение яркостью.
3. Отбуксируйте тег в место размещения в данной инструкции.



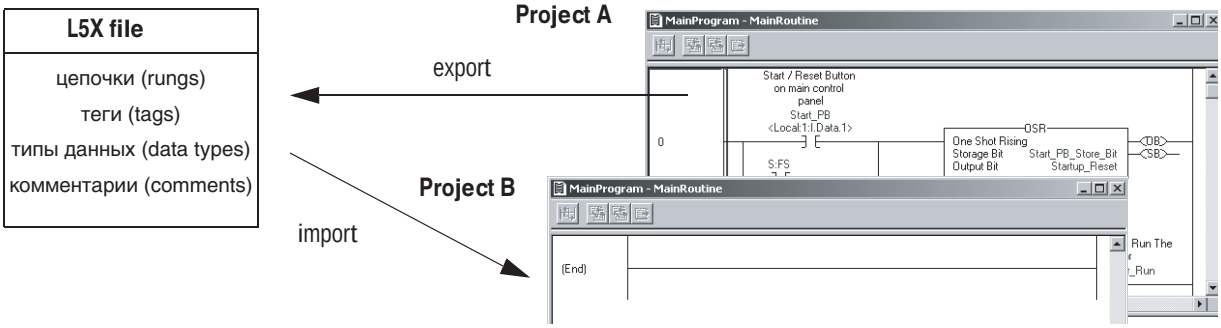
Присвоение непосредственного значения (постоянной)

1. Щелкните в области операнда данной инструкции.
2. Ведите значение и нажмите [Enter].

Экспорт/импорт релейной логики

Если вы хотите повторно использовать релейную логику из другого проекта, просто экспортируйте ее в файл L5X, а затем импортируйте в нужный проект. Файл L5X содержит все, что необходимо для релейной логики, за исключением модулей ввода/вывода.

RSLogix 5000 software 13.0 or later



Если вы импортируете цепочки

Когда вы импортируете цепочки, программное обеспечение RSLogix 5000 показывает список тегов и типов данных, заданных пользователем, расположенные вдоль цепочек. Используйте этот список для управления тегами и типами данных при операции импортирования.

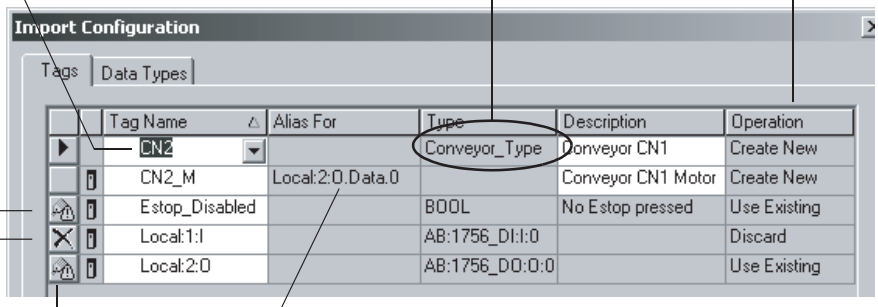
В колонка Operation (Операция) показывается, что случается с каждым тегом и типом данных при импортировании. Программное обеспечение либо создает их, либо использует существующие в проекте, либо отбрасывает (не импортирует).

Если это необходимо, вы можете переименовать тег, чтобы он лучше соответствовал данному проекту.

Если вы размещаете переменные для цепочек в тип данных, определяемый пользователем, то у вас меньше тегов для управления.

Если тег уже существует в проекте, вы можете:

- использовать существующий тег, который отключит тег в файле библиотеки и привяжет алгоритм к существующему тегу.
- переименовать тег, который создает новый тег.



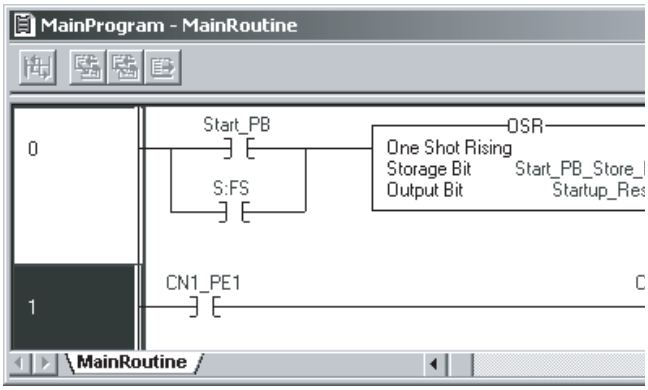
Новый тег ввода/вывода (I/O) не создается.

Если тег ввода/вывода уже существует в проекте, операция импортирования использует этот тег для всех псевдонимов для этого имени тега. Когда вы импортируете проект, убедитесь, что вы проверили псевдонимы тегов.

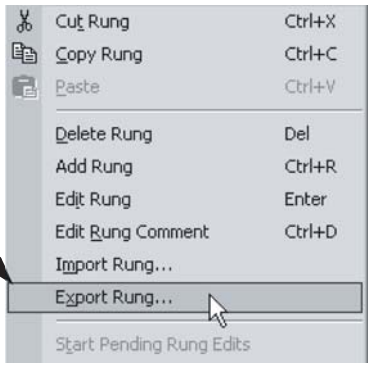
Экспорт цепочек

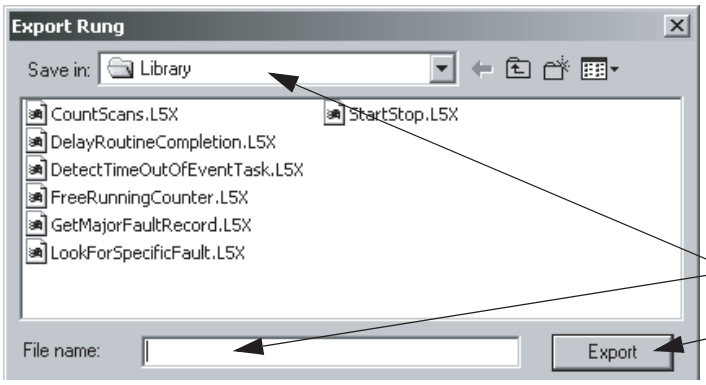
1. Выберите цепочки для экспорта.

Если цепочки:	То:
Последовательны	Щелкните на первой цепочке и удерживая [Shift] щелкните на последней цепочке.
Не последовательны	Щелкните на первой цепочке и удерживая [Ctrl] щелкните на каждой дополнительной цепочке.



2. Щелкните правой клавишей мыши и выберите *Export Rung (Цепочка для экспорта)*.

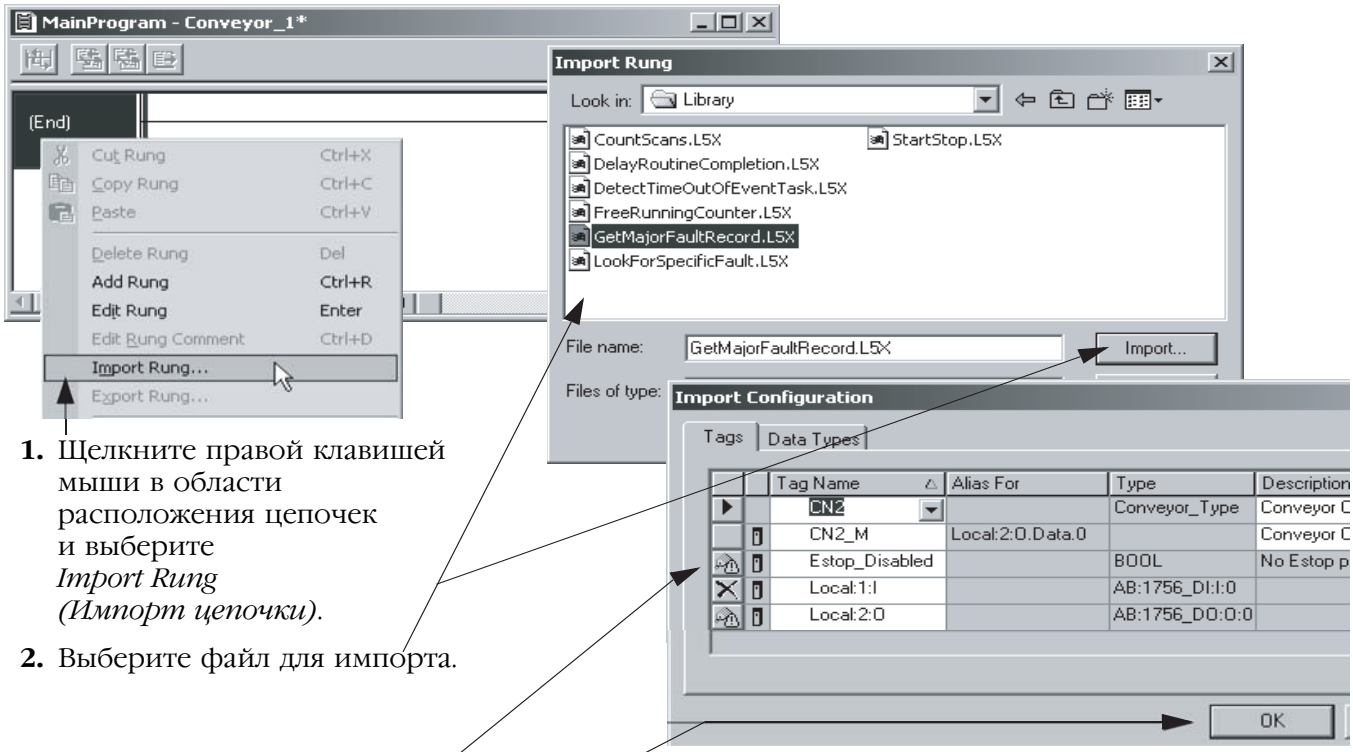




3. Выберите расположение и имя файла

4. Создайте файл.

Импорт цепочек



1. Щелкните правой клавишей мыши в области расположения цепочек и выберите *Import Rung (Импорт цепочки).*

2. Выберите файл для импорта.

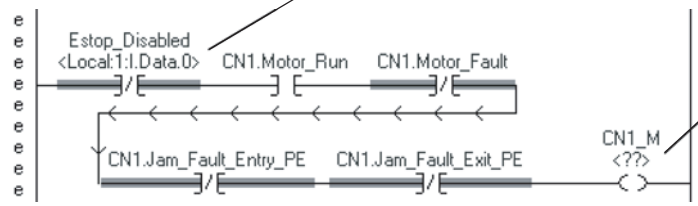
3. Проверьте конфликтность имен.

4. Импортируйте файл.

Проверка псевдонимов тегов

Если вы импортируете псевдоним тега, убедитесь, что он указывает на правильный базовый тег. Если тег является псевдонимом для тега, уже существующего в проекте, программное обеспечение устанавливает взаимосвязь между псевдонимом и базовыми тегами.

Импортируемые цепочки



Если проект не имеет базового тега, вы должны либо создать базовый тег, либо указать на другой базовый тег.

Проверка процедуры

При программировании процедуры (процедур), периодически проверяйте свою работу:

1. В самой верхней линейке инструментов окна RSLogix 5000 щелкните на  .
2. Если внизу окна будут перечислены какие-либо ошибки:
   a. Перейдите на первую ошибку или предупреждение и нажмите [F4].
   b. Исправьте ошибку в соответствии с описанием в окне Results (Результаты).
   c. Вернитесь к пункту 1.
3. Чтобы закрыть окно Results (Результаты), нажмите [Alt] + [1].

Для заметок:

Программирование на языке функциональных блоков

Когда используется эта процедура

Используйте эту процедуру для следующих целей:

- Организации процедур на языке функциональных блоков,
- Создания одной или более схем функциональных схем для процедуры,
- Ввода функциональных схем в процедуру.

Перед использованием данной процедуры

Перед началом использования данной процедуры убедитесь, что у вас есть разрешение на выполнение следующих задач:

☐ Navigate the Controller Organizer (Осуществлять навигацию контроллера)

☐ Identify the Programming Languages That Are Installed (Идентифицировать установленные языки программирования)

За более подробной информацией относительно данных задач обратитесь к разделу «Начало» (Getting Started) на стр. 1-1.

Как использовать данную процедуру

При программировании процедур на языке функциональных блоков выполняйте следующие этапы:

☐ Идентификация листов для процедуры

☐ Выбор элементов функциональных блоков

☐ Выбор имени тега для элемента

☐ Задание порядка выполнения

☐ Идентификация коннекторов

☐ Задание режима управления (программный/оператор)

☐ Добавление листа

☐ Добавление элемента функционального блока

☐ Соединение элементов

☐ Присваивание тега

☐ Присваивание непосредственного значения (постоянной)

☐ Соединение блока с OCON и ICON

☐ Проверка процедуры

Идентификация листов для процедуры

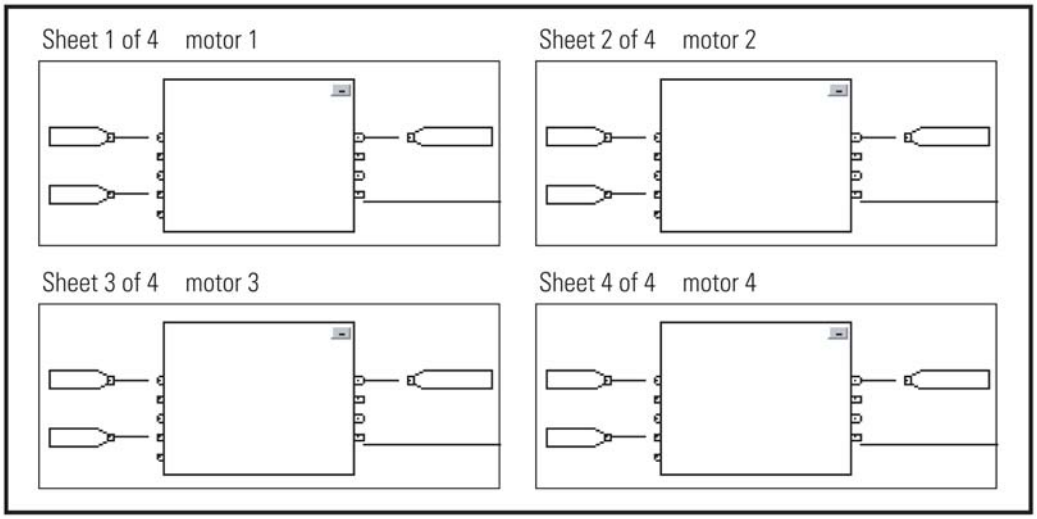
Для упрощения перемещения по процедуре на языке функциональных блоков разделите процедуру на ряд листов.

- Использование листов позволит вам организовать размещение функциональных блоков и облегчить их поиск. Использование листов не оказывает влияния на порядок выполнения блоков.
- При выполнении процедуры выполняются все листы
- Общей рекомендацией является использование одного листа для одного прибора (мотора, клапана и т.д.).

В следующем ниже примере показана процедура управления четырьмя двигателями.

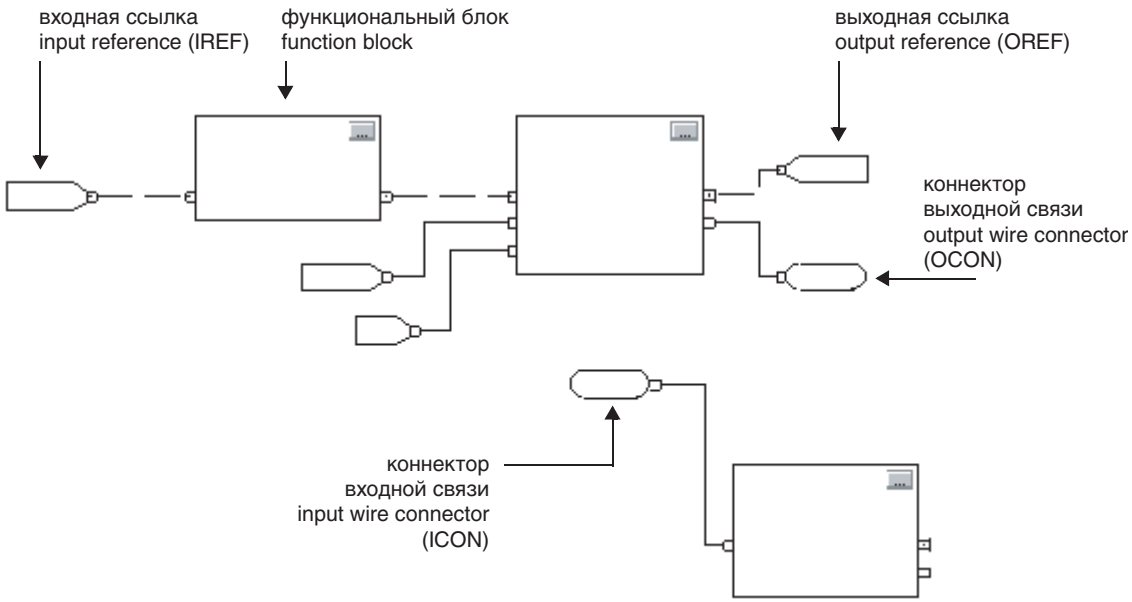
ПРИМЕР Процедура управления двигателем

Процедура управления двигателем



Выбор элементов функционального блока

Для управления устройством используйте следующие элементы:



При выборе элементов функционального блока используйте следующую таблицу:

Если вы хотите:	Используется:
получить величину из устройства ввода или тега	входная ссылка (input reference (IREF))
отправить величину в выходное устройство или тег	выходная ссылка (output reference (OREF))
выполнить операцию над входной величиной или величинами и выдать выходную величину или величины	функциональный блок (function block)
передавать данные между функциональными блоками когда они: • на большом расстоянии друг от друга на одном листе • на разных листах в пределах одной процедуры	выходной коннектор связи (output wire connector (OCON)) и входной коннектор связи (input wire connector (ICON))
рассредоточить данные по нескольким точкам в одной программе	один выходной коннектор связи (OCON) и несколько входных коннекторов связи (ICON)

Выбор имени тега для элемента

Каждый функциональный блок использует тег для хранения конфигурации и информации о состоянии для данной инструкции.

- Когда вы добавляете процедуру на языке функциональных блоков, программное обеспечение RSLogix 5000 автоматически создает для этого блока тег. Вы можете использовать этот тег как он есть, переименовать этот тег или присвоить другой тег.
- Для IREF и OREF вы можете создать тег или присвоить существующий.

В следующей ниже таблице представлены форматы для имени тега:

Для:	Задайте:
Тега	*tag_name*
Битовый номер наибольшего типа данных	*tag_name.bit_number*
Элемента конструкции	*tag_name.member_name*
Элемента одномерного массива	*tag_name[x]*
Элемента двухмерного массива	*tag_name[x,y]*
Элемента трехмерного массива	*tag_name[x,y,z]*
Элемента массива в пределах конструкции	*tag_name.member_name[x]*
Члена элемента массива	*tag_name[x,y,z].member_name*

Где:

x позиция элемента для первой размерности,

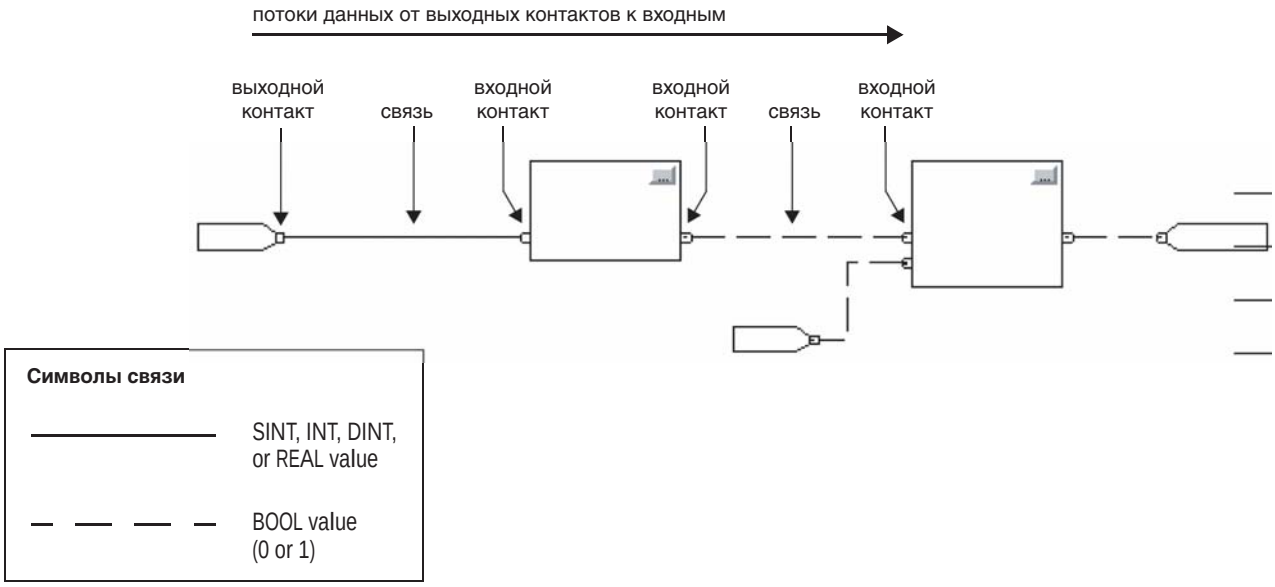
y позиция элемента для второй размерности,

z позиция элемента для третьей размерности.

Для конструкции в конструкции дополнительно добавьте *.member_name* .

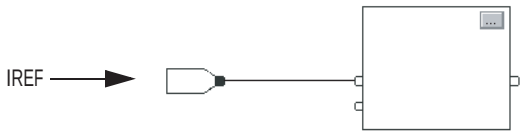
Порядок выполнения

Вы определяете порядок выполнения (поток данных) связывая функциональные блоки между собой и указывая, если это необходимо, обратные связи. Месторасположение функциональных блоков не влияет на порядок, в котором они выполняются.

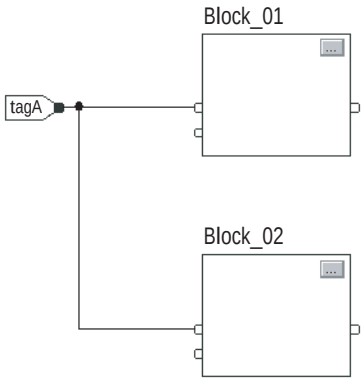


Фиксация данных

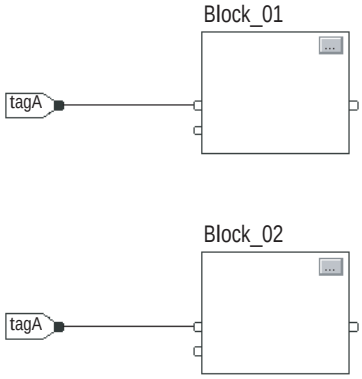
Если вы используете элемент IREF для того, чтобы задать исходные данные для какого-нибудь функционального блока, данные в этом элементе «защелкиваются» для сканирования подпрограммы этого функционального блока. Элемент IREF «защелкивает» данные для тегов программы и контроллера. Контроллер обновляет все данные в IREF при начале каждого сканирования.



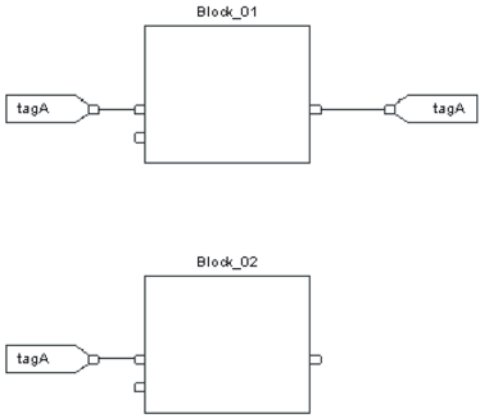
В данном примере значение тега tagA запоминается в начале выполнения процедуры. Это сохраненное значение используется при выполнении блока Block_01. Это же самое значение используется при выполнении блока Block_02. Если значение tagA изменяется во время выполнения процедуры, значение tagA в элементе IREF не изменяется до следующего выполнения процедуры.



Этот пример аналогичен представленному выше. Значение tagA запоминается один раз в начале выполнения процедуры. Процедура использует это значение внутри себя.



Начиная с 11 версии программного обеспечения RSLogix 5000, вы можете использовать один и тот же тег в нескольких элементах IREF и одном элементе OREF в одной процедуре. Поскольку значения тегов в элементах IREF защелкиваются при каждом сканировании процедуры, все IREF будут использовать одно и то же значение, даже если OREF будет получать различные значения тегов при выполнении этой процедуры. В данном примере, если tagA имеет значение 25.4, то когда эта программа начинает выполняться, это значение сканируется и Block_01 изменяет значение tagA на 50.9, второй элемент IREF подключенный к Block_02 будет по-прежнему использовать значение 25.4 при выполнении Block_02 на этом сканировании. Новое значение tagA равное 50.9 не будет использоваться никаким элементом IREF в данной программе, пока не начнется следующее сканирование.



Порядок выполнения

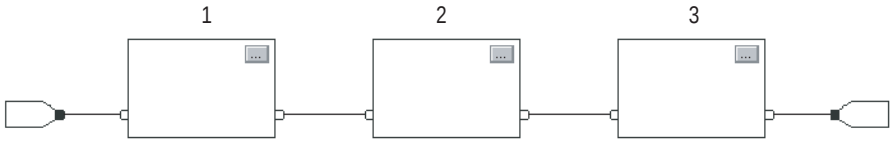
Программное обеспечение RSLogix 5000 автоматически определяет порядок выполнения функциональных блоков в программе когда вы:

- проверяете процедуру с функциональными блоками,
- проверяете проект, содержащий процедуру на языке функциональных блоков,
- загружаете проект, содержащий процедуру на языке функциональных блоков.

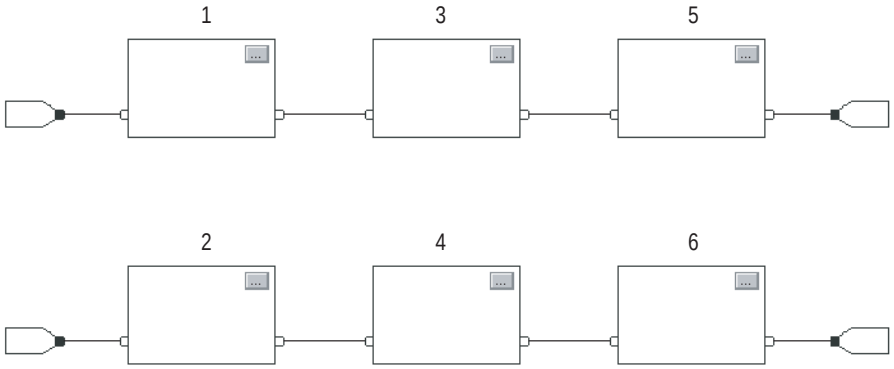
Вы определяете порядок выполнения, связывая функциональные блоки между собой и указывая, если это необходимо, поток данных обратных связей.

Если функциональные блоки не связаны друг с другом, то не имеет значения, какой блок выполняется первым. Между этими блоками нет потока данных.

Если вы связали блоки последовательно, порядок выполнения идет от входа к выходу. Входы блока требуют наличия данных до того, как этот блок начнет выполняться. Например, block 2 должен выполняться раньше блока block 3, потому что block 2 предоставляет входные данные для блока block 3.



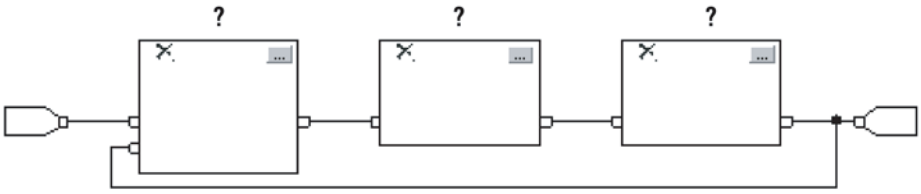
Порядок выполнения блоков имеет значение только для блоков, связанных друг с другом. Следующий ниже пример справедлив, поскольку две группы блоков не связаны друг с другом. Блоки в пределах заданной группы выполняются в последовательности, определяемой их положением внутри этой группы.



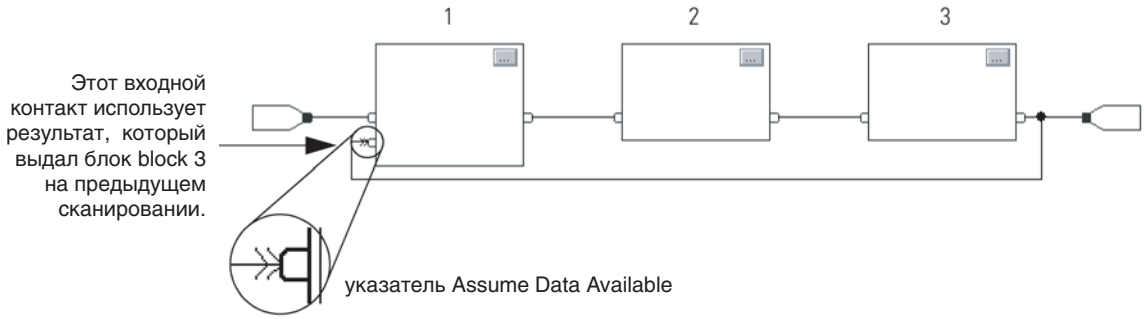
Организация циклов

Для создания цикла обратной связи свяжите выводной контакт блока с входным контактом того же блока. Пример, представленный ниже, справедлив. Цикл содержит один единственный блок и порядок выполнения роли не играет.

Если группа блоков объединена в цикл, то контроллер не может определить какой блок выполнять первым. Другими словами, он не может разрешить цикл.

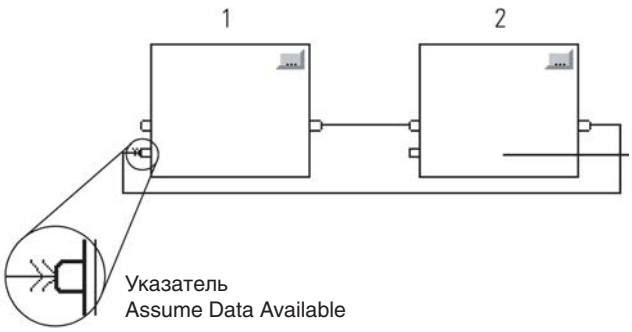
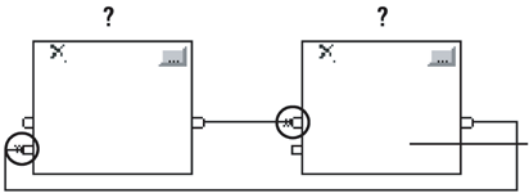


Чтобы определить блок, который будет выполняться первым, пометьте входной контакт, который создает цикл (обратная связь), при помощи указателя *Assume Data Available* (допустим, что данные имеются). В следующем ниже примере блок block 1 использует результат блока 3, который был получен на предыдущем сканировании программы.



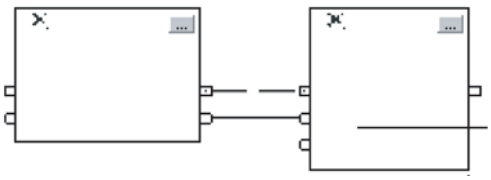
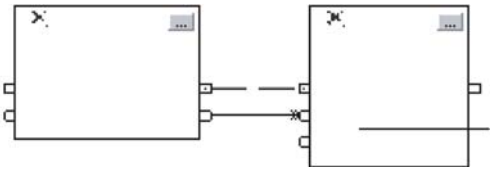
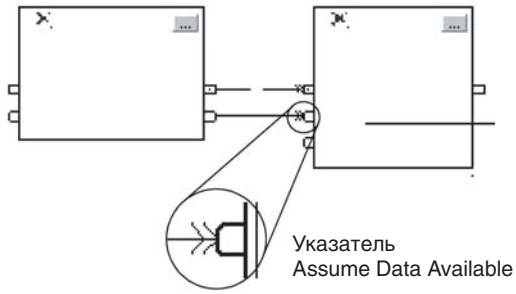
Указатель *Assume Data Available* определяет поток данных внутри цикла. Стрелка указывает, что эти данные служат входом для первого блока цикла.

Не помечайте все связи цикла указателем *Assume Data Available.*

Это правильно	**Это неправильно**
 Указатель *Assume Data Available* определяет поток данных внутри цикла.	 Контроллер не может разрешить цикл, поскольку все связи используют указатель *Assume Data Available*.

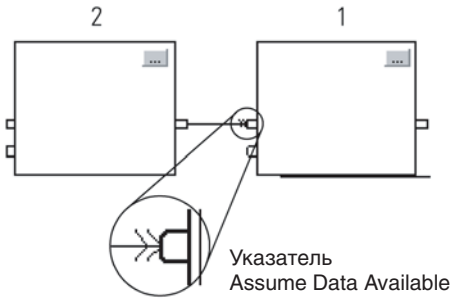
Разрешение потока данных между двумя блоками

Если вы используете две или более связи между двумя блоками, используйте одинаковые указатели потока данных для всех связей между двумя блоками.

Это правильно	Это неправильно
 Ни та, ни другая связь не используют указатель *Assume Data Available*.	 Одна связь использует указатель *Assume Data Available*, в то время как другая нет.
 Обе связи используют указатель *Assume Data Available*.	

Создание задержки на одно сканирование

Чтобы организовать задержку на одно сканирование между блоками, используйте указатель *Assume Data Available*. В следующем ниже примере блок block 1 выполняется первым. Он использует выход блока, который получается на предыдущем сканировании процедуры.



Резюме

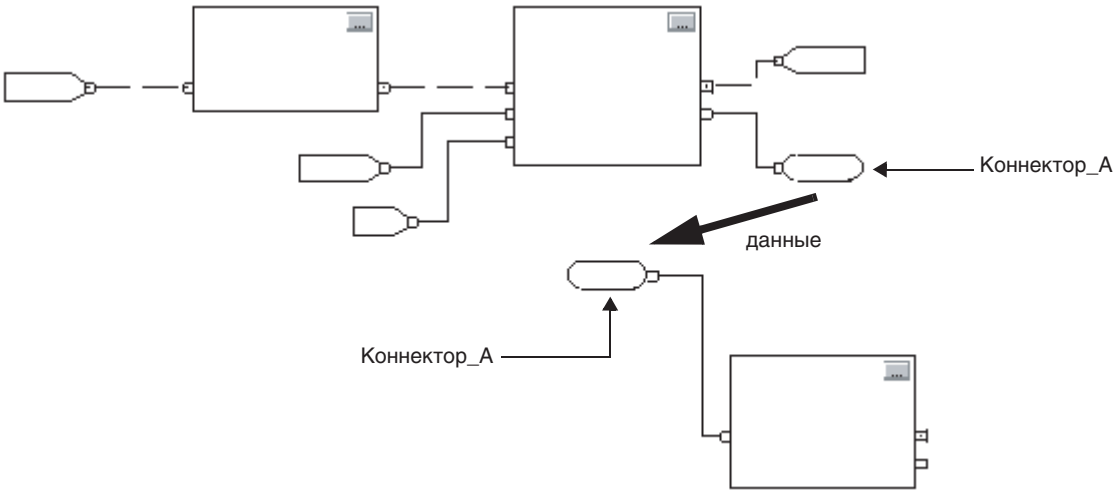
В итоге, процедура на языке функциональных блоков выполняется следующим образом:

1. Контроллер «защелкивает» все данные в элементах IREF.
2. Контроллер выполняет функциональные блоки в той последовательности, в которой они связаны.
3. Контроллер записывает результаты в элементы OREF.

Идентификация коннекторов

Как и связи, коннекторы передают потоки данных от выходных контактов к входным контактам. Используйте коннекторы когда:

- Элементы, которые вы хотите соединить, расположены на разных листах в одной процедуре,
- Размещение на релейной схеме связи затруднено ввиду наличия других связей или элементов
- Вы хотите разделить распределить данные по нескольким точкам в процедуре.



При использовании коннекторов следуйте следующим правилам:

- Каждый OCON требует уникального имени.
- Для каждого OCON вы должны иметь, по крайней мере, один ICON (т.е. ICON с тем же именем, что и OCON).
- Несколько ICON могут ссылаться на один OCON. Это позволяет вам распределять данные по нескольким точкам в процедуре.

Задание режима управления (программный/оператор)

Несколько инструкций поддерживают режимы управления программный/оператор.

- Эти инструкции:
- Улучшенный выбор (Enhanced Select (ESEL)
- Сумматор (Totalizer (TOT)
- Улучшенный ПИД (PIDE)
- Линейное измерение/выдержка (RMPS)
- Дискретное 2-х режимное устройство (D2SD)
- Дискретное 3"х режимное устройство (D3SD)

Способ управления программа/оператор позволяет вам управлять этими инструкциями одновременно как из пользовательской программы, так и при помощи интерфейса оператора. При программном управлении (Program) инструкция управляется при помощи программных входов (Program inputs) для этой инструкции, а при управлении оператором (Operator) инструкция управляется входами оператора (Operator inputs).

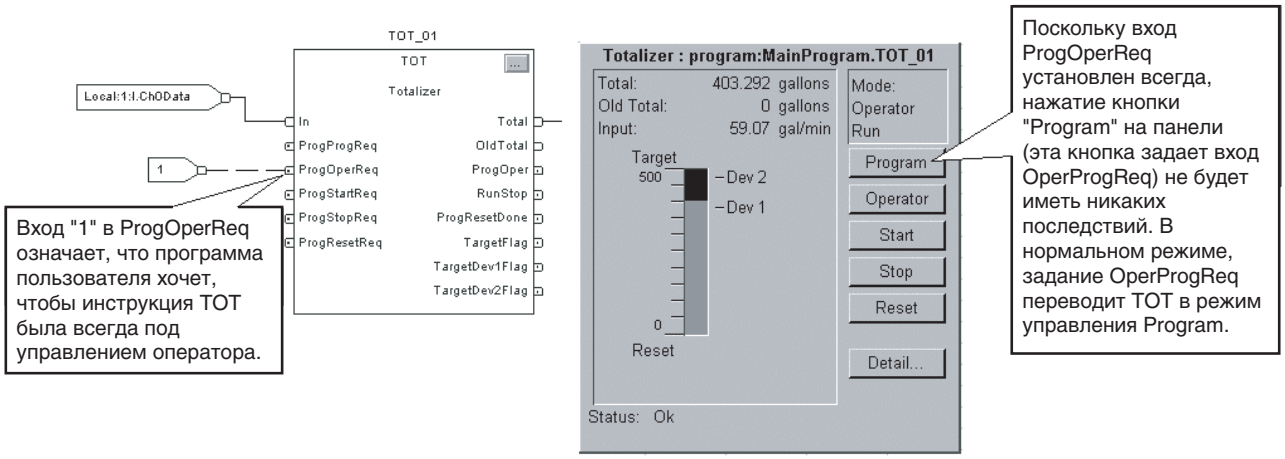
Режимы управления Program или Operator задаются при помощи следующих входов:

Вход:	Описание:
.ProgProgReq	Программный запрос на переход к программному управлению (Program).
.ProgOperReq	Программный запрос на переход к управлению оператором (Operator).
.OperProgReq	Запрос оператора на переход к программному управлению (Program).
.OperOperReq	Запрос оператора на переход к управлению оператором (Operator).

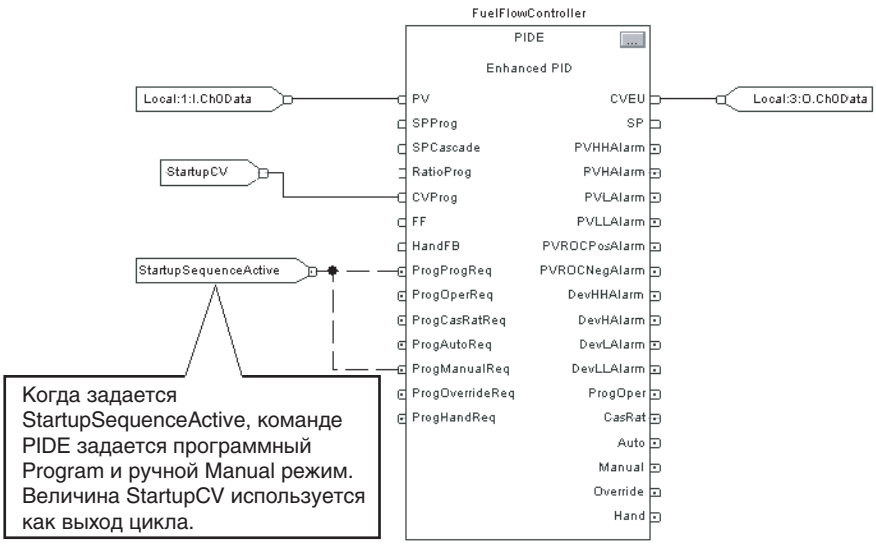
Для того чтобы определить, в каком режиме управления находится инструкция (Program или Control), проверьте выходное значение ProgOper. Если ProgOper установлен, инструкция находиться в режиме управления Program; если ProgOper сброшено, инструкция находится в режиме управления Operator.

Если установлены оба бита запроса режима управления, то режим Operator имеет приоритет над режимом Program. Например, если оба бита ProgProgReq и ProgOperReq установлены, инструкция переходит в режим управления Operator.

Входы программных запросов (Program request) имеют приоритет над входами запросов оператора (Operator request). Это дает возможность использовать входные значения ProgProgReq и ProgOperReq для «фиксации» инструкции в желаемом режиме управления. Например, допустим, что инструкция Totalizer будет всегда работать под управлением оператора, и ваша программа никогда не сможет управлять запуском или остановом инструкции Totalizer. В этом случае вы должны ввести (подсоединить) литерную величину 1 в ProgOperReq. Это не даст возможности оператору перевести инструкцию Totalizer в программный режим управления, задав OperProgReq с интерфейса оператора.



Аналогично, постоянная установка ProgProgReq может «зафиксировать» инструкцию в программном режиме управления. Это полезно при последовательном включении, когда вы хотите, чтобы программа управляла работой инструкции не опасаясь, что оператор, по невнимательности, возьмет управление над инструкцией. В этом примере программа задает вход ProgProgReq при запуске, а затем очищает ProgProgReq, когда запуск завершен. После того как ProgProgReq сброшен, инструкция останется в программном режиме управления, пока не получит запрос на его изменение. Например, оператор может задать OperOperReq с панели для того, чтобы взять управление инструкцией в свои руки. В следующем примере показано как «зафиксировать» инструкцию в программном режиме управления.



Входные значения запроса оператора в инструкцию всегда сбрасываются инструкцией при выполнении. Это позволяет интерфейсам оператора работать с этими инструкциями, просто задавая бит запроса желаемого режима. Вам не нужно программировать интерфейс оператора на сброс битов запроса. Например, если интерфейс оператора задает вход OperAutoReq для инструкции PIDE, то когда выполняется инструкция PIDE, она определяет, каким должен быть соответствующий отклик, и сбрасывает OperAutoReq.

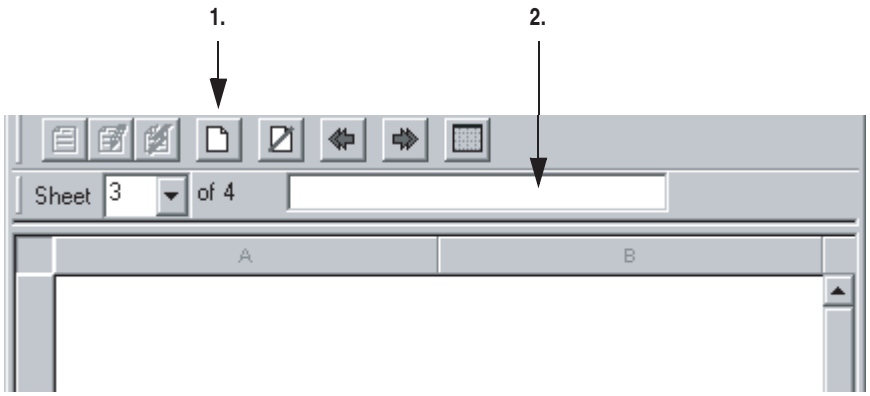
Входные значения программных запросов обычно не сбрасываются инструкцией, поскольку они задаются как входные связи. Если инструкция сбросит эти входные значения, они будут восстановлены элементом входной связи. Может возникнуть ситуация, когда вы захотите задать программные запросы (Program requests) таким образом, чтобы они сбрасывались инструкцией. В этом случае вы можете задать вход ProgValueReset и программа будет всегда сбрасывать значения программных запросов при выполнении.

В этом примере цепочка релейной логики в другой процедуре используется для разового защелкивания ProgAutoReq для инструкции PIDE при нажатии на пусковую кнопку. Поскольку инструкция PIDE автоматически сбрасывает программные запросы, у вас нет необходимости записывать какой либо алгоритм релейных схем для того, чтобы сбрасывать ProgAutoReq после выполнения процедуры, и инструкция будет получать только один запрос на переход в режим Auto при каждом нажатии кнопки.

Когда TIC101AutoReq Pushbutton нажата, происходит разовое защелкивание ProgAutoReq для инструкции PIDE TIC101. TIC101 уже была сконфигурирована с заданным входным значением ProgValueReset так, что когда инструкция PIDE выполняется, она автоматически сбрасывает ProgAutoReq.

Добавление листа

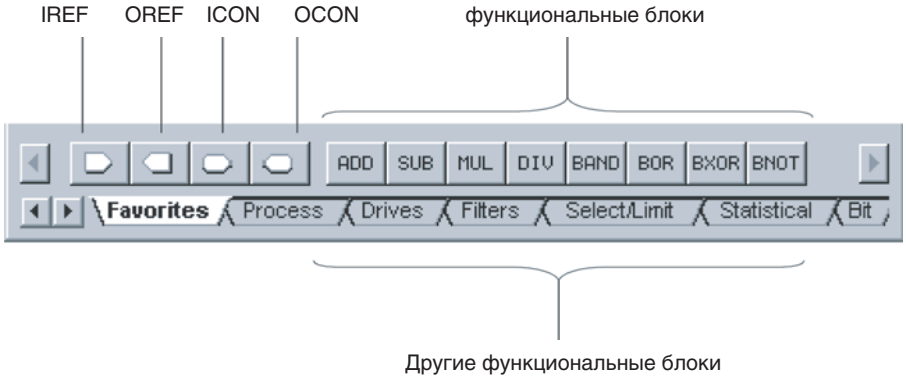
Чтобы добавить лист к процедуре на языке функциональных блоков сделайте следующее:



1. Щелкните на  .
2. Введите описание листа (до 50 символов).

Добавление элемента на языке функциональных блоков

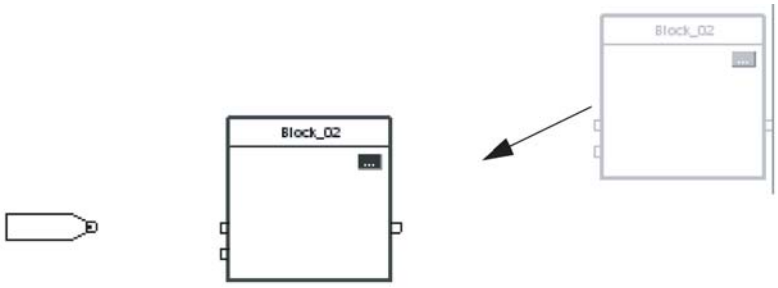
Для добавления элемента на языке функциональных блоков используйте панель инструментов Language Element (Элементы языка).



Чтобы добавить элемент:

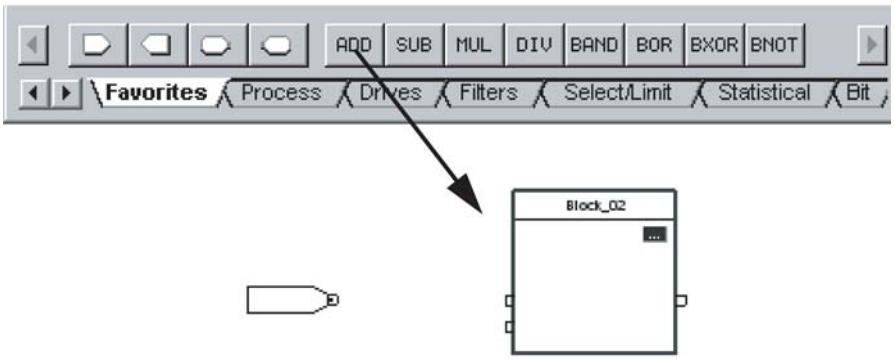
1. На панели инструментов Language Element (Элементы языка) щелкните на кнопке того элемента, который вы хотите добавить.
2. Отбуксируйте этот элемент в нужное место.

   Например



Вы можете так же отбуксировать кнопку донного элемента в нужное место.

Например



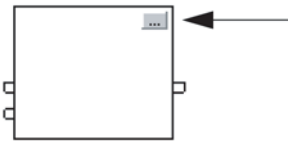
Соединение элементов

Для того чтобы определить поток данных, используются опции:

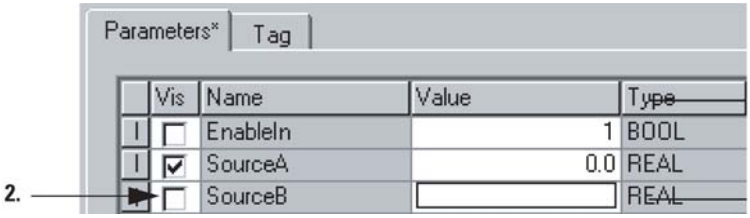
- ☐ Показать или скрыть контакт (Show or Hide a Pin)
- ☐ Связать элементы между собой (Wire Elements Together)
- ☐ Пометить связь указателем Assume Data Available (Mark a Wire with the Assume Data Available Indicator).

Показать или скрыть контакт

Когда вы добавляете инструкцию на языке функциональных блоков, ниже показана настройка по умолчанию для контактов параметров. Остальные контакты являются скрытыми. Чтобы показать или скрыть контакт:



1. Щелкните на кнопке  блока.



2. Очистите или установите флажок *Vis* контакта:

Если вы хотите:	**То:**
Спрятать контакт	Снимите флажок *Vis.*
Показать контакт	Установите флажок *Vis.*

3. Выберите  .

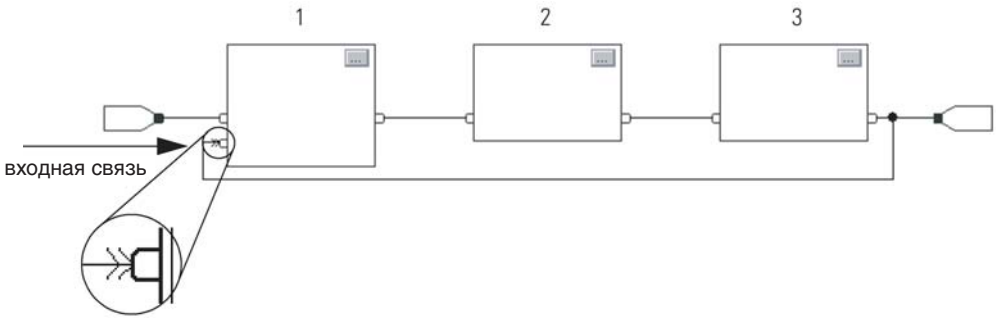
Связать элементы между собой

Чтобы связать два элемента между собой, щелкните на выходном контакте первого элемента, а затем щелкните на входном контакте другого элемента. Зеленая точка указывает подходящую точку связи.

Например

Пометить связь указателем Assume Data Available

Чтобы пометить связь как входную, щелкните правой клавишей мыши и выберите *Assume Data Available*.



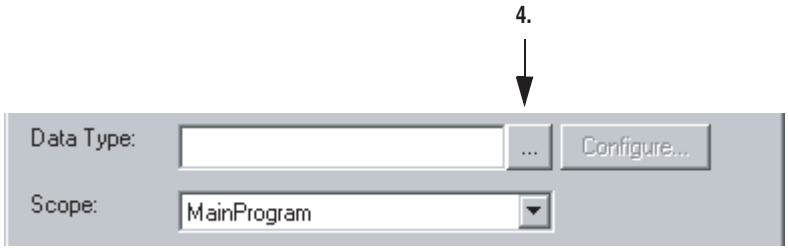
Присваивание тега

Для присваивания тега элементу у вас имеются следующие опции:

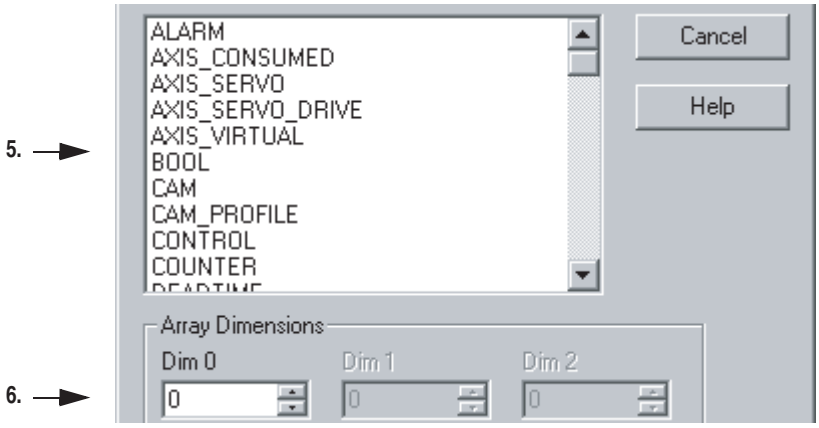
- ☐ Создать и присвоить новый тег (Create and Assign a New Tag)
- ☐ Переименовать тег функционального блока (Rename the Tag of a Function Block)
- ☐ Присвоить существующий тег (Assign an Existing Tag)

Создание и присвоение нового тега

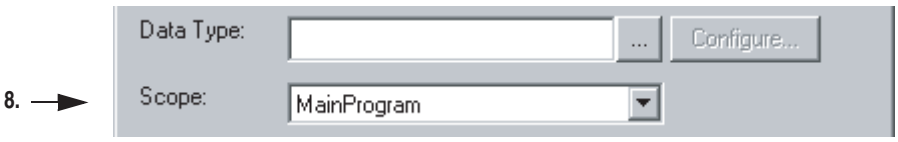
1. Дважды щелкните в области операнда.
2. Введите имя тега и нажмите [Enter].
3. Щелкните правой клавишей мыши на имени тега и выберите *New "tag_name" (Новое имя тега)*.



4. Щелкните на кнопке [...] .



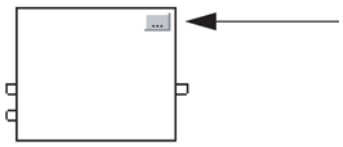
5. Выберите для этого тега тип данных.
6. Если вы хотите определить тег как массив, введите число элементов для каждой размерности.
7. Выберите [OK] .



8. Выберите для тега область применимости.

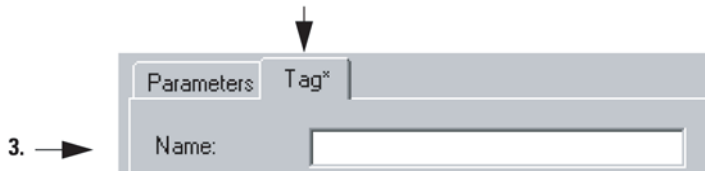
9. Выберите OK .

Переименовать тег функционального блока



1. Щелкните на кнопке ... данного блока.

2. Щелкните на закладке *Tag (Тег)*.



3. Введите имя нового тега для данного блока.

4. Выберите OK .

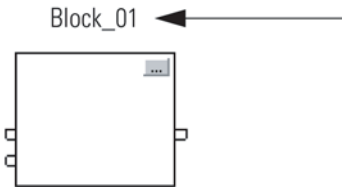
Присвоить существующий тег

1. Дважды щелкните в области операнда.

2. Щелкните на ▼

3. Выберите имя:

Чтобы выбрать:	Сделайте:
тег	Дважды щелкните на имени тега
Номер бита	А. Щелкните на имени тега. Б. Справа от имени тега щелкните на ▼ В. Щелкните на нужном бите.



4. Нажмите [Enter] или щелкните в другом месте схемы.

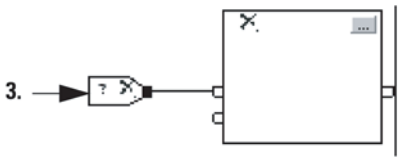
Присвоение непосредственного значения (постоянной)

Для того, чтобы входному параметру присвоить постоянное значение вместо значения тега, мы можете использовать следующие опции:

Если вы хотите:	То:
Сделать значение видимым на схеме или отчетах	Используйте IREF
Иметь возможность изменять значение в режиме он-лайн без редактирования процедуры	Введите Value (Значение) в Tag of a Block (Тег блока)

Использование IREF

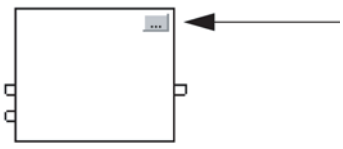
1. Добавьте IREF.
2. Свяжите IREF с входным контактом, который получает данное значение.



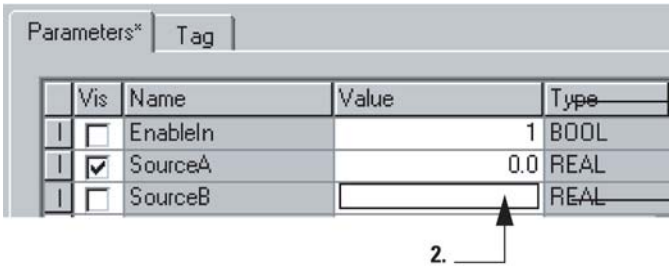
3. Дважды щелкните в области операнда IREF.
4. Введите значение и нажмите *Enter*.

Ввод значения (Value) в тег блока (Tag of a Block)

Чтобы присвоить значение параметру, когда связь подключается к контакту:



1. Щелкните на кнопке [...] блока.



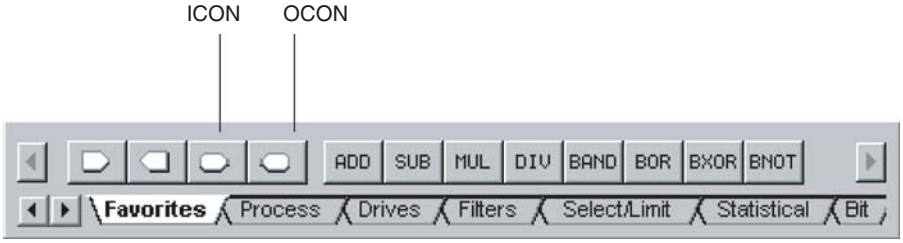
2. Введите значение.
3. Выберите OK .

Соединение блоков при помощи OCON и ICON

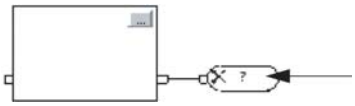
Для передачи данных между листами или в случае сложных ситуаций при организации связи:

☐ Добавьте OCON

☐ Добавьте ICON

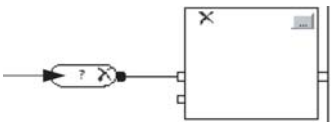


Добавление OCON



1. Добавьте коннектор выходной связи (OCON) и разместите его около выходного контакта, который предоставляет данное значение.
2. Соедините OCON с этим выходным контактом.
3. Дважды щелкните в области операнда этого OCON.
4. Введите имя, которое идентифицирует этот коннектор и нажмите [Enter].

Добавление ICON



1. Добавьте коннектор входной связи (ICON) и разместите его около входного контакта, который получает значение от соответствующего OCON.
2. Соедините ICON с этим входным контактом.
3. Дважды щелкните в области операнда этого ICON.
4. Выберите имя OCON, который передает значение этому коннектору и затем щелкните на пустой области данной схемы.

Проверка процедуры

По мере того, как вы программируете процедуру, периодически проверяйте свою работу:

1. В самой верхней линейки инструментов окна RSLogix 5000 щелкните на  .
2. Если внизу окна будут перечислены какие-либо ошибки:
   a. Перейдите на первую ошибку или предупреждение и нажмите [F4].
   b. Исправьте ошибку в соответствии с описанием в окне Results (Результаты).
   c. Вернитесь к пункту 1.
3. Чтобы закрыть окно Results (Результаты), нажмите [Alt] + [1].

Связь с другими устройствами

Как использовать данную главу

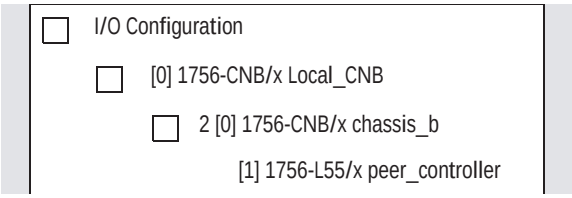
Информация, представленная в данной главе, будет полезна при организации связи между модулями ввода/вывода контроллера и другими контроллерами.

За информацией о:	Обращайтесь на стр.:
Соединениях	10-1
Производящем и потребляющем теге	10-9
Выполнении инструкции Message (MSG) (Сообщение)	10-19
Получении или настройке количества неподключенных буферов	10-25
Преобразования типов INT и DINT	10-28

Соединения

Контроллер Logix5000 использует **соединения** (connections) для многих, но не для всех связей с другими устройствами.

Термин:	Определение:
Соединение	Коммуникационная связь между двумя устройствами, такая как связь между контроллером и модулем ввода\вывода, терминалом PanelView и другим контроллером. Соединения – это ресурсы, для обеспечения более надежной связи между устройствами, чем обмен сообщениями без соединения. Косвенно, вы определяете количество соединений, используемых контроллерами при конфигурировании контроллера для связи с другими устройствами данной системы. Соединения используются следующими типами связи: • модулями ввода\вывода, • производящими и потребляющими тегами, • определенными типами инструкций Message (MSG)

Термин:	Определение:
Требуемый пакетный интервал (RPI)	RPI задает период обновления данных через соединение. Например, модуль ввода посылает данные в контроллер в соответствии с RPI, который вы приписываете данному модулю. • Обычно RPI задается в миллисекундах. Диапазон от 0.2 (200 микросекунд) до 750 миллисекнд. • Если сеть ControlNet соединяет заданные устройства, RPI резервирует участок в потоке в потоке данных, проходящих через сеть ControlNet. Расчет времени для этого участка может не совпадать с точным значением RPI, но система управления гарантирует, что данные будут передаваться с частотой, заданной RPI.
путь (path)	Путь описывает маршрут, реализуемый соединением до точки назначения. Обычно вы автоматически определяете путь для соединения, когда добавляете устройство к папке I/O Configuration (Конфигурация ввода\вывода) контроллера. 

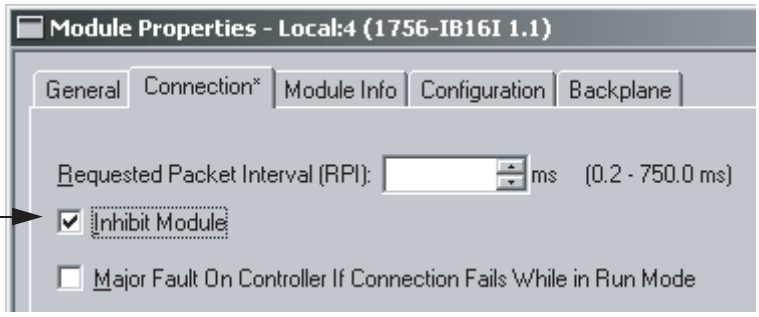
Блокировка соединения

ВНИМАНИЕ

Блокировка модуля разрывает соединение с этим модулем и предотвращает обмен данными ввода\вывода.

В некоторых ситуациях, таких как, например, ввод в эксплуатацию системы, полезно ограничивать возможности системы управления, и вновь вводить их по мере подключения системы управления. Контроллер позволяет блокировать отдельные модули или группы модулей, что останавливает контроллер от попыток связаться с этими модулями.

Блокировка связи с этим модулем.



Когда вы конфигурируете модуль ввода\вывода, по умолчанию он деблокирован. Вы можете изменить свойства отдельного модуля, чтобы блокировать его.

Если вы хотите:	**То:**
иметь возможность связи с модулем	не блокируйте модуль
блокировать возможность связи с модулем	блокируйте модуль

Когда вы блокируете модуль коммуникационного моста, такого как 1756-CNB или 1756-DHRIO, контроллер выключает соединения с этим мостом и со всеми модулями, которые зависят от этого моста. Блокировка коммуникационного моста позволяет вам выключить целую ветвь сети ввода\вывода.

Когда вы устанавливаете флажок блокировки модуля, контроллер выводит на экран значок ⚠ поверх этого модуля.

Если вы:	**И:**	**И:**	**То:**
автономны	→	→	Состояние блокировки сохраняется в проекте. Когда в загружаете проект, модуль остается блокированным.
работаете в системе	блокируете модуль, когда вы соедининены с этим модулем	→	Соединение с этим модулем закрывается. Выходы модулей переходят к последнему сконфигурированному программному режиму.
	Блокируете модуль, но соединение с этим модулем не было установлено (может быть из-за условия ошибки или сбоя)	→	Этот модуль блокируется. Информация о состоянии модуля меняется и указывает, что модуль блокирован, а не поврежден.
	Разблокируете модуль (снимая флажок)	нет сбоя	С модулем устанавливается соединение и этот модуль автоматически динамически реконфигурируется (если этот контроллер является контроллером-владельцем) при помощи конфигурации, которую вы создали для этого модуля. Если контролер сконфигурирован только на ожидание сигнала, он не может реконфигурировать модуль.
		имеет место сбой	Соединение с модулем не устанавливается. Информация о состоянии модуля изменяется, указывая на условие сбоя.

Чтобы блокировать или разблокировать модуль по отношению к алгоритму:

1. Используйте инструкцию Get System Value (GSV) (Получить системное значение) для считывания атрибута Mode (Режим) данного модуля.
2. Установите или очистите бит 2:

Если вы хотите:	То:
Блокировать модуль	Установите бит 2
Разблокировать модуль	Очистите бит 2

3. Используйте инструкцию Set System Value (SSV) (Настроить системное значение) для записи атрибута Mode (Режим) для данного модуля.

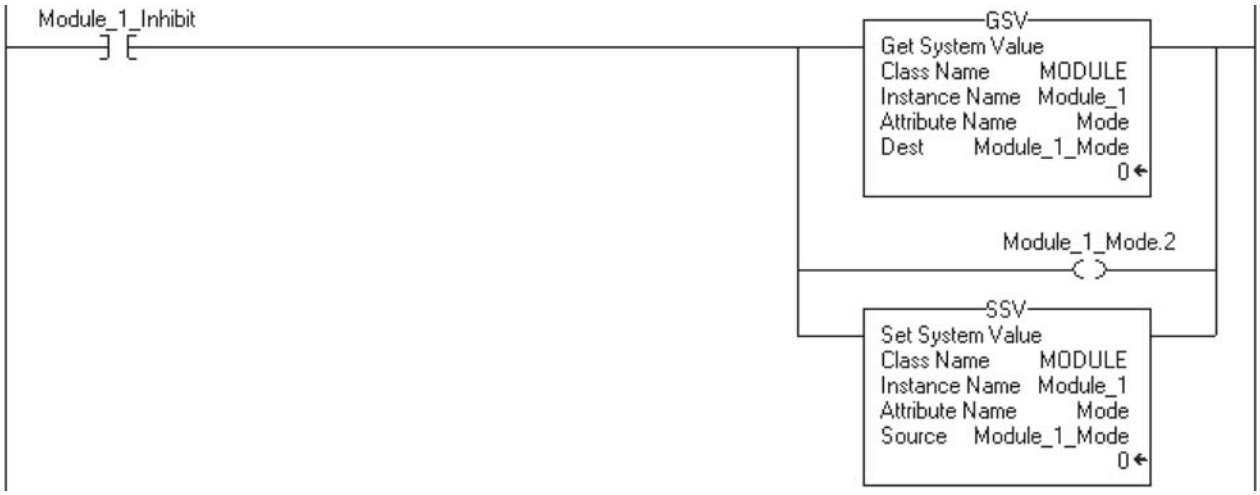
ПРИМЕР Блокировка соединения

Когда *Module_1_Inhibit* = 1, блокируйте операцию ввода\вывода модуля с именем *Module_1*:

1. Инструкция GSV устанавливает *Module_1_Mode* = значению атрибута Mode для этого модуля.

2. Инструкция OTE устанавливает бит 2 *Module_1_Mode* = 1. Это означает блокировку соединения.

3. Инструкция SSV устанавливает атрибут Mode для модуля = *Module_1_Mode*.



Обработка ошибок связи

ВНИМАНИЕ

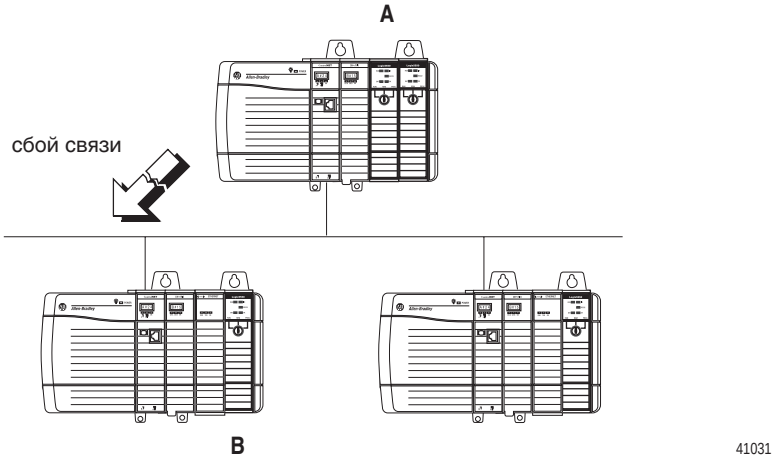
Выходы отвечают последнему состоянию входов (которое имело место до сбоя). Для того, чтобы избежать возможных травм и повреждения оборудования, убедитесь, что при этом не возникают опасные операции. Сконфигурируйте важные модули ввода\вывода таким образом, чтобы контроллер генерировал основную ошибку при потере их соединений с контроллером. Или отслеживайте состояние модулей ввода\вывода.

Если контроллер теряет связь с модулем, данные от этого модуля не обновляются. Если такое случается, логика принимает решение относительно данных, которые могут быть правильными, а могут и таковыми и не быть.

ПРИМЕР

Потеря связи

Контроллер **B** запрашивает данные у контроллера **A**. Если связь между контроллерами обрывается, то контроллер B продолжает действовать в соответствии с последними данными, которые он получил от контроллера A.



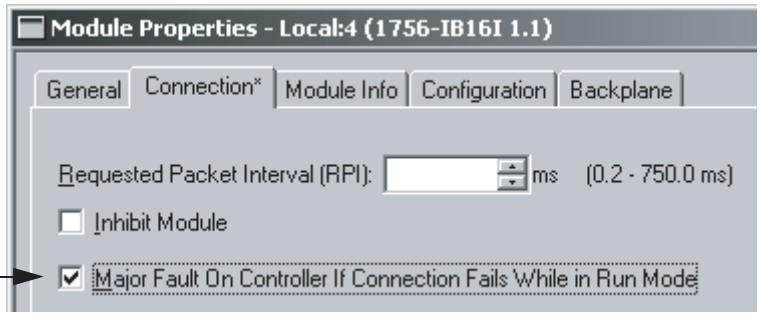
Если связь с устройством из конфигурации ввода\вывода контроллера отсутствует в течение 100 миллисекунд, связь прерывается по истечению времени ожидания. Если это случается, вы можете воспользоваться следующими возможностями:

Если вы хотите чтобы контроллер:	То:
выдал ошибку (основную)	сконфигурируйте основную ошибку
продолжил работу	отслеживайте состояние модуля

Конфигурирование основной ошибки

Вы можете сконфигурировать модули так, чтобы в контроллере генерировалась основная ошибка при потере связи с этим контроллером. Это прерывает выполнение релейной логики и запускает обработчик ошибок контроллера (Controller Fault Handler). Если обработчик ошибок не снимает эту ошибку, контроллер выключается.

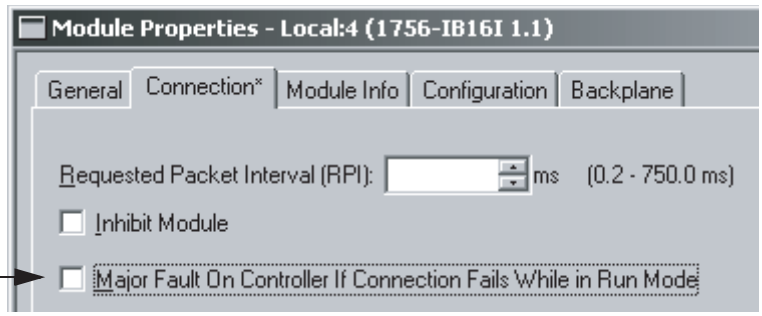
Если имеет место превышение времени ожидания, в контроллере генерируется основная ошибка



Отслеживание состояние модуля

Если вы не сконфигурировали основную ошибку, вы можете отслеживать состояние модуля. Если модуль теряет связь с контроллером, выходные значения принимают значения, заданные для случая сбоя. Контроллер и другие модули ввода\вывода продолжают работать, основываясь на старых данных от этого модуля.

Если имеет место превышение времени ожидания, продолжайте работу без генерации основной ошибки в контроллере



Если для связи превышается время ожидания, контроллер выдает следующие предупреждения:

- Индикатор I/O LED на передней панели контроллера мигает зеленым.
- Появляется символ ⚠ на папке конфигурации ввода\вывода и устройствах, для которых превышено время ожидания.
- генерируется код ошибки модуля, к которому имеется доступ при помощи:
  - окна Module Properties (Свойства модуля) этого модуля
  - инструкции GSV.

Для отслеживания состояния соединений, используйте инструкцию Get System Value (GSV) (Получение системного значения) для контроля объекта MODULE в интересах контроллера или заданного модуля:

Если вы хотите:	Получить этот атрибут	Тип данных	Описание	
Определить, имеет ли место превышение времени ожидания с каким-либо устройством	LEDStatus	INT Для большей производительности используйте тип DINT как целевой тип данных	Задает текущее состояние индикатора I/O LED на передней панели контроллера. **Примечание:** Вам необязательно вводить имя экземпляра с этим атрибутом. Этот атрибут применяется к совокупности модулей.	
			Величина	**Значение**
			0	**LED off** (отключен) Нет объектов MODULE, сконфигурированных для контроллера (нет модулей в разделе I/O Configuration (конфигурация ввода\вывода) программы-секретаря контроллера).
			1	Мигающий красный : Нет объектов MODULE в режиме выполнения.
			2	Мигающий зеленый: По крайней мере, один объект MODULE не в режиме выполнения.
			3	Немигающий зеленый: все объекты Module в режиме выполнения.
Определить, имеет ли место превышение времени ожидания с заданным устройством	FaultCode	INT Для большей производительности используйте тип DINT как целевой тип данных	Номер, который идентифицирует сбой модуля, если таковой имеет место. В Instance Name выберите устройство, соединение с которым вы хотите контролировать. Убедитесь, что этому устройству приписано имя в папке I/O Configuration (Конфигурация ввода\вывода) данного проекта.	

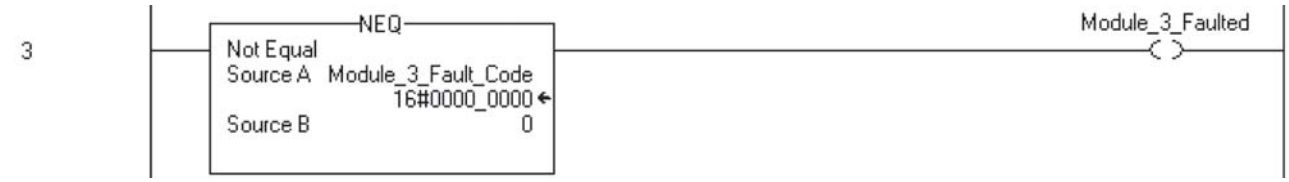
ПРИМЕР

Отслеживание состояния модуля.

Инструкция GSV непрерывно устанавливает *I_O_LED_Status* (тег DINT) = состоянию индикатора контроллера I/O LED.

Если *I_O_LED_Status* = 2, то связь имеет превышение времени ожидания (сбой), по крайней мере, для одного модуля. Инструкция GSV устанавливает *Module_3_Fault_Code* = коду сбоя для *Module_3*.

Если *Module_3_Fault_Code*. не равен 0, имеет место превышение времени ожидания (сбой) с *Module_3*. Инструкия OTE задает *Module_3_Faulted* = 1.

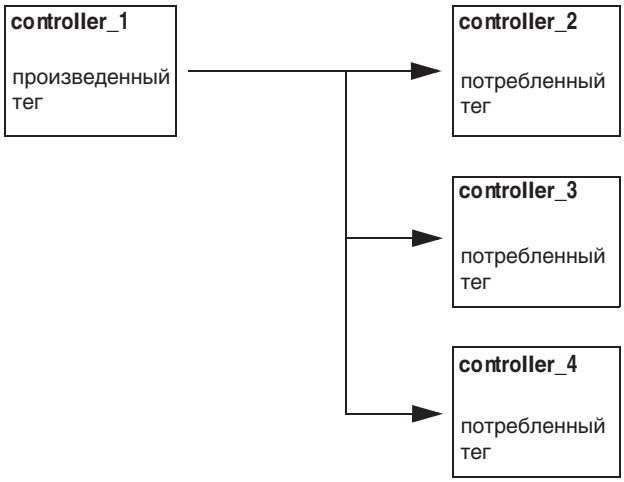


Производящий и потребляющий теги

У вас имеются следующие опции для передачи данных между контроллерами (отправки или получения данных):

Если данные:	То:	См.
Необходимо регулярно доставлять с определенным вами интервалом. (детерминистически)	Произведите и потребите тэг	Данный раздел
Осуществлять отправку по наступлении какого-либо события в вашем приложении.	Выполните инструкцию Execute a Message (MSG)	Стр. 10-19

Контроллер Logix5000 позволяет вам производить (рассылать) и потреблять (получать) системно-обобщенные теги.



Термин:	Определение:
Произведенный тег	Тег, который контроллер делает доступным для использования другими контроллерами. Несколько контроллеров могут одновременно потреблять (получать) эти данные. Произведенный тег посылает свои данные одному или более тегам потребителям без использования логики. Произведенный тег посылает свои данные в соответствии с RPI потребляющего тега.
Потребленный тег	Тег, который получает данные от производящего тега. Тип данных потребляющего тега должен совпадать с типом данных производящего тега (включая размерность массивов). RPI потребляющего тега определяет период обновления данных.

Контроллеры и сети поддерживающие возможность производства и потребления тегов

Следующая ниже таблица используется для поиска комбинаций, которые позволят воспользоваться возможностью производства и потребления тегов.

Этот контроллер	Может производить и потреблять теги в сети:		
	Backplane	ControlNet	EtherNet/IP
SLC 500		x	
PLC-5		x	
CompactLogix(1)			x
ControlLogix	x	x	x
DriveLogix			
FlexLogix		x	
SoftLogix58		x	

(1) Используйте контроллер CompactLogix5335, номер по каталогу 1769-L35E.

Чтобы два контроллера совместно производили или потребляли теги, оба контроллера должны быть подключены к одной сети (такой, так ControlNet или Ethernet/IP network). Вы не можете воспользоваться мостом для работы в двух сетях.

Требования соединения для производства и потребления тегов

.

ВАЖНО Если соединение для потребления тега обрывается, все остальные теги, являющимися потребителями от этого удаленного контроллера, перестают получать новые данные.

Производство и потребление тегов требуют наличия соединения. Когда вы увеличиваете количество контроллеров, потребляющих произведенные теги, вы также уменьшаете количество соединений, доступных контроллеру для других операций, таких как связи или ввод/вывод.

Каждый произведенный или потребленный тег использует следующее количество соединений:

Этот контроллер:	И этот тип тега:	Используют данное количество соединений:
ControlLogix SoftLogix5800	Произведенный тег	*number_of_consumers + 1*
	Потребленный тег	*1*
CompactLogix DriveLogix FlexLogix	Произведенный тег	*number_of_consumers*
	Потребленный тег	*1*

ПРИМЕР

Требования со стороны соединения для произведенного или потребленного тэга (Produced or Consumed Tag):

- Контроллер FlexLogix производящий тэг для 5 контроллеров (потребителей) использует 5 соединений.
- Контроллер ControlLogix производящий 4 тэга для 1 контроллера использует 8 соединений:
  - каждый тэг использует 2 соединения (1 потребитель +1 =2).
  - 2 соединения на тэг x 4 тега = 8 соединений.
- Потребление 4x тэгов от контроллера использует 4 соединения (1 соединение на тег x 4 тега =4 соединения).

Организация тегов для производства и потребления данных

При организации создании тэгов, которые будут, в конечном счете, производить или потреблять данные (данные для совместного использования), следуйте следующим руководящим принципам:

<table>
<tr><th>Указание:</th><th colspan="4">Подробности:</th></tr>
<tr><td>Создайте тэг в области данных контроллера.</td><td colspan="4">Вы можете совместно использовать теги только из области данных контроллера.</td></tr>
<tr><td>Используйте один из этих типов данных:
• DINT;
• REAL;
• массив DINT или REAL;
• заданный пользователем.</td><td colspan="4">• Для совместного использования других типов данных, создайте тип данных, определяемый пользователем, который содержит требующиеся данные.
• Используйте одинаковый тип данных для произведенного тэга и для соответствующего потребленного тега(ов).</td></tr>
<tr><td rowspan="6">Для совместного использования тэгов с контроллером PLC 5C используйте тип данных, определяемый пользователем.</td><td>Для:</td><td>Этого:</td><td colspan="2">То:</td></tr>
<tr><td rowspan="3">производства</td><td>целых</td><td colspan="2">Создайте тип данных, задаваемый пользователем, который содержит массив INT с четным количеством элементов, такой как INT[2]. (Когда вы производите тип INT, вы должны производить два или более.)</td></tr>
<tr><td>Только одного значения REAL</td><td colspan="2">Используйте тип данных REAL.</td></tr>
<tr><td>Более чем одного значения REAL</td><td colspan="2">Создайте тип данных, определяемый пользователем, который содержит массив REAL.</td></tr>
<tr><td colspan="2">Потребления целого типа</td><td colspan="2">Создайте тип данных, определяемый пользователем, который содержит следующие члены:
Тип данных: — Описание:
DINT — Состояние (Status)
INT[x], где x выходной размер данных от контроллера PLC-5C. (Если вы потребляете только тип INT, то опустите x.) — Данные, произведенные контроллером PLC-5C.</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Ограничьте размер тэга до размера менее или равного 500 байт.</td><td colspan="4">• Если вы должны передать более 500 байт, создайте логику для передачи данных в пакетах. Смотрите главу 11.
• Если вы производите тэг в сети ControlNet, может понадобиться тэг менее 500 байт. Смотрите раздел «Настройка пределов пропускной способности» на странице 10-13.</td></tr>
<tr><td>Используйте наивысшее возможное значение RPI для вашего приложения</td><td colspan="4">Если контроллер потребляет тег в сети ControlNet, используйте двоично кратное время обновления в сети ControlNet (NUT). Например, если NUT 5 миллисекунд, используйте RPI в 5, 10, 20, 40 мс, и тд.</td></tr>
<tr><td>Сгруппируйте данные для одного контроллера</td><td colspan="4">Если вы производите несколько тэгов для одного контроллера:
• Сгруппируйте данные в один или более тип данных, определяемых пользователем. (Это уменьшит количество соединения, по сравнению со случаем создания каждого тега отдельно.)
• Сгруппируйте данные в зависимости от скорости обновления. (Для преобразования пропускной способности сети, используете больший RPI для менее критичных данных.)
Например, вы можете создать один тэг для критичных данных и другой тэг для не критичных данных.</td></tr>
</table>

Настройка пределов пропускной способности

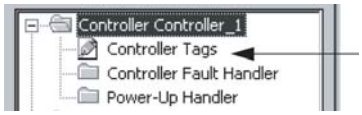
Когда вы совместно используете тэг в сети ControlNet, тэг должен соответствовать пределам пропускной способности данной сети:

- Когда количество соединений в сети ControlNet увеличивается, несколько соединений, включающих произведенные или потребленные тэги, могут потребовать объединения времени обновления сети (NUT).
- Поскольку сеть ControlNet может передавать только 500 байт в один NUT, данные каждого соединения должны быть меньше 500 байт, чтобы соответствовать NUT.

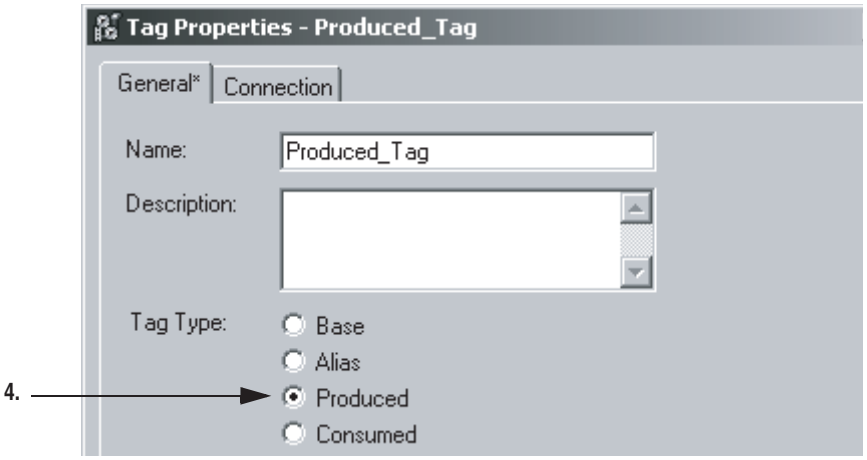
В зависимости от размера вашей системы вы может не иметь достаточной пропускной способность вашей сети ControlNet для тэга 500 бай. Если тэг слишком велик для вашей сети ControlNet, сделайте одну или более из следующих настроек:

<table>
<tr><th>Настройка:</th><th colspan="2">Описание:</th></tr>
<tr><td>Уменьшите время обновления сети (NUT).</td><td colspan="2">Для более быстрых NUT меньшее количество соединений должны объединять интервал обновления.</td></tr>
<tr><td>Увеличьте затребованный интервал пакета (RPI) ваших соединений.</td><td colspan="2">При высоких RPI, соединения могут чередовать посылку данных во время интервала обновления.</td></tr>
<tr><td rowspan="4">Для модуля моста (CNB) сети ControlNet в удаленном шасси
Выберите наиболее формат связи для этого шасси:</td><td>Является большинство модулей данного шасси недиагностируемыми, цифровыми модулями ввода/вывода?</td><td>Затем выберите формат связи для удаленного модуля CNB:</td></tr>
<tr><td>Да</td><td>Оптимизация стойки (рэка) (Rack Optimization)</td></tr>
<tr><td>Нет</td><td>Нет</td></tr>
<tr><td colspan="2">Формат оптимизации стойки (рэка) (Rack Optimization) использует дополнительно 8 байт для каждого слота в своем шасси. Аналоговые модули или модули, которые посылают или получают диагностические, временные, запланированные данные или данные о предохранителях требуют прямого соединения и не могут иметь преимущества от формата
оптимизации рэка. Выберите «None» для освобождения до 8 байт на
слот для другого использования, такого как производство или потребление тегов.</td></tr>
<tr><td>Разделите тег на два или более мелких тэгов:</td><td colspan="2">1. Сгруппируйте данные для одинаковых времен обновления. Например, вы можете создать один тэг для критичных данных и другой тэг для данных, которые не критичны.
2. Присвойте различные RPI каждому тэгу.</td></tr>
<tr><td>Создайте логику для передачи данных меньшими порциями (пакетами).</td><td colspan="2">Смотрите «Создание больших массивов» на странице 11-1.</td></tr>
</table>

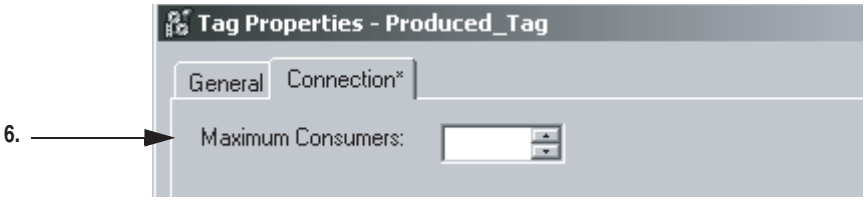
Производство тега



1. Откройте проект RSLogix 5000, содержащий тэг, который вы хотите сделать производящим.
2. В программе-организаторе контроллера щелкните правой клавишей мыши на папке *Controller Tags (Теги контроллера)*и выберите *Edit Tags (Редактирование тегов)*. (Вы можете делать производящие тэги только в области данных контроллера).
3. В окне Controller Tags (Теги контроллера) щелкните правой клавишей мыши на теге, который вы хотите сделать производящим, и выберите *Edit Tag Properties (Редактировать свойства тега)*.



4. Выберите кнопку опции *Produced (Производящий)*.
5. Щелкните на закладке *Connection (Соединение)*.

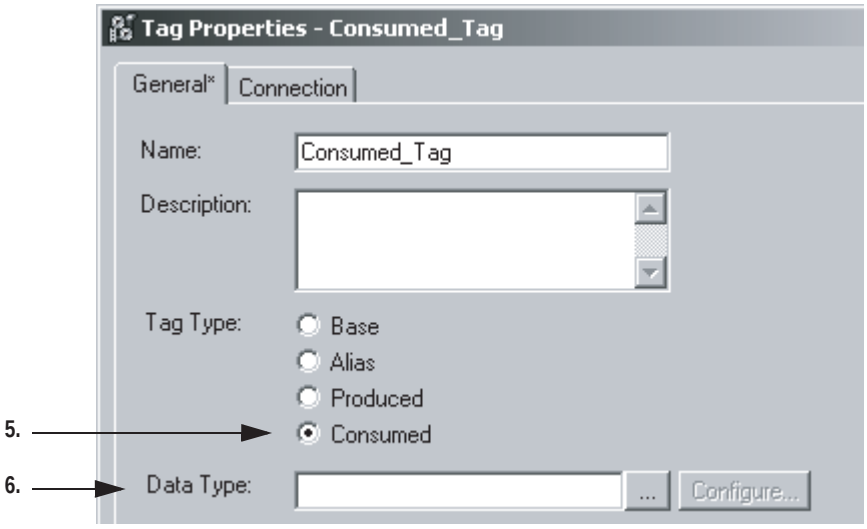


6. Введите или выберите количество контроллеров, которые будут потреблять (получать) этот тэг.
7. Щелкните на OK .

Потребление данных, произведенных другим контроллером

1. Откройте проект RSLogix 5000 который будет потреблять данные.
2. В программе-организаторе контроллера, папке *I/O Configuration* добавьте контроллер, который производит данные (другой контроллер Logix5000 или PLC-5C).

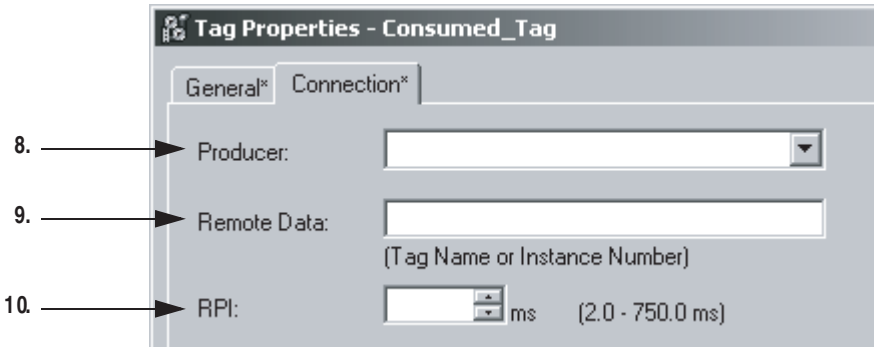
3. В программе-организаторе контроллера щелкните правой клавишей мыши на папке *Controller Tags (Теги контроллера)* и выберите *Edit Tags (Редактирование тегов)*. (Только тэги из области данных контроллера могут потреблять данные).
4. В окне Controller Tags (Теги контроллера) щелкните правой клавишей мыши на теге, который будет потреблять данные, и выберите *Edit Tag Properties (Редактировать свойства тега)*.



5. Выберите кнопку опции *Consumed* (Потребленный).
6. Убедитесь, что данные имеют следующий тип:

Если производящий контроллер:	**Тип данных должен быть:**	
Logix5000	Тот же тип данных, что и для производящего тега	
PLC-5C	Тип данных, определяемый пользователем, который содержит следующие члены:	
	Тип данных:	**Описание:**
	DINT	Состояние (Status)
	INT[*x*], где *x* выходной размер данных от контроллера PLC-5C. (Если вы потребляете только одно INT, то опустите *x.)*	Данные, произведенные контроллером PLC-5C.

7. Щелкните на закладке *Connection* (Соединение).



8. Выберите контроллер, производящий данные.

9. Введите имя или номер экземпляра производимых данных.

Если производящий контроллер:	**То введите или выберите:**
Logix5000	Имя тега для производящего тега
PLC-5C	Номер сообщения от конфигурации ControlNet контроллера PLC-5C

10. Введите или выберите требуемый пакетный интервал (RPI) для данного соединения.

11. Выберите OK .

12. Если вы потребляете тег в сети ControlNet, используйте программное обеспечение RSNetWorx для планирования сети.

Дополнительные шаги для контроллера PLC-5C

Если вы используете данные совместно с контроллером PLC-5C, выполните следующие дополнительные действия:

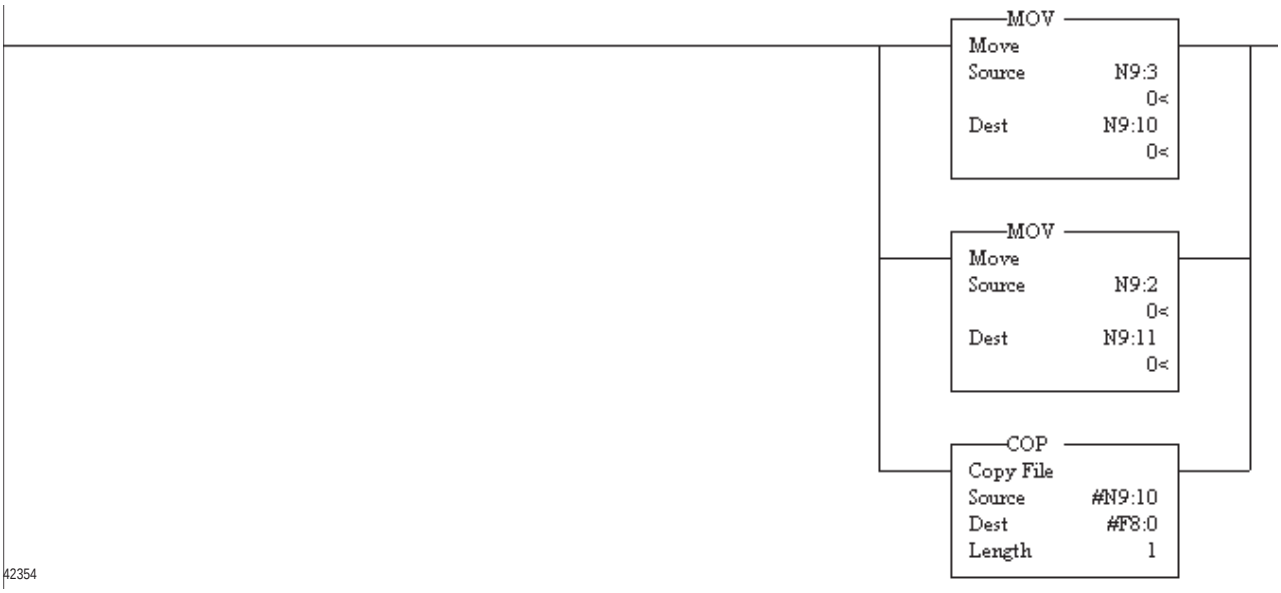
<table>
<tr><th>Действие:</th><th colspan="3">Подробности:</th></tr>
<tr><td rowspan="4">В конфигурации ControlNet контроллера PLC-5C запланируйте сообщение</td><td>Если PLC-5C</td><td>Это:</td><td>Тогда при помощи RSNetWorx</td></tr>
<tr><td>производит</td><td>целые</td><td>В конфигурации ControlNet контроллера PLC-5C вставьте Send Scheduled Message (Отправить запланированное сообщение).</td></tr>
<tr><td rowspan="2">потребляет</td><td>целые</td><td>В конфигурации ControlNet контроллера PLC-5C:
A. Вставьте Receive Scheduled Message (Получить запланированное сообщение).
B. В Message size (Размер сообщения) введите количество целых значений в произведенном теге.</td></tr>
<tr><td>вещественные</td><td>В конфигурации ControlNet контроллера PLC-5C:
A. Вставьте Receive Scheduled Message (Получить запланированное сообщение).
B. В Message size (Размер сообщения) введите удвоенное количество вещественных значений в произведенном теге. Например, если произведенный тег содержит 10 вещественных значений, введите 20 для Message size (размер сообщения).</td></tr>
<tr><td>Если контроллер PLC-5C потребляет REAL, реконструируйте значения.</td><td colspan="3">Если вы производите вещественное значение (32 разрядное значение с плавающей точкой) для контроллера PLC-5C, PLC-5C сохраняет эти данные в последовательных 16-разрядных целых значениях.
• Первое целое содержит верхние разряды (самые левые).
• Второе целое содержит младшие разряды (самые правые).
• Такой шаблон остается неизменным для всех значений с плавающей точкой.
См. пример на стр. 10-18.</td></tr>
</table>

Следующий ниже пример показывает, как реконструировать вещественные значения (с плавающей точкой) в контроллере PLC-5C.

ПРИМЕР

Реконструкция значений с плавающей точкой

Две инструкции MOV меняют местами целые значения, когда целые перемещаются в новое положение. Поскольку назначением инструкции COP является адрес с плавающей точкой, то она берет два последовательных целых, всего 32 бита, и конвертирует их в одно значение с плавающей точкой.



Выполнение инструкции Message (MSG) (Сообщение)

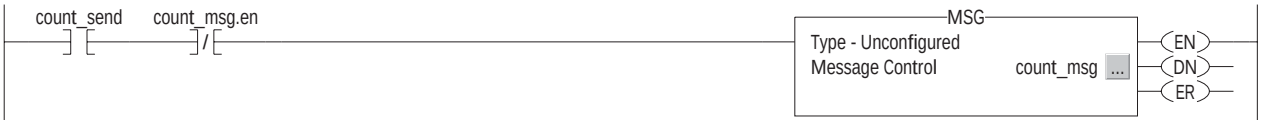
Для передачи данных между контроллерами (отправка или получение сообщений) у вас есть следующие возможности:

Если данные:	То:	См.
Необходимо регулярно доставлять данные с установленной вами скоростью (детерминистически)	Произведите и потребите тэг	Стр. 10-9
Отправляются, когда имеет место заданное условие	Выполните инструкцию Execute a Message (MSG)	Данный раздел

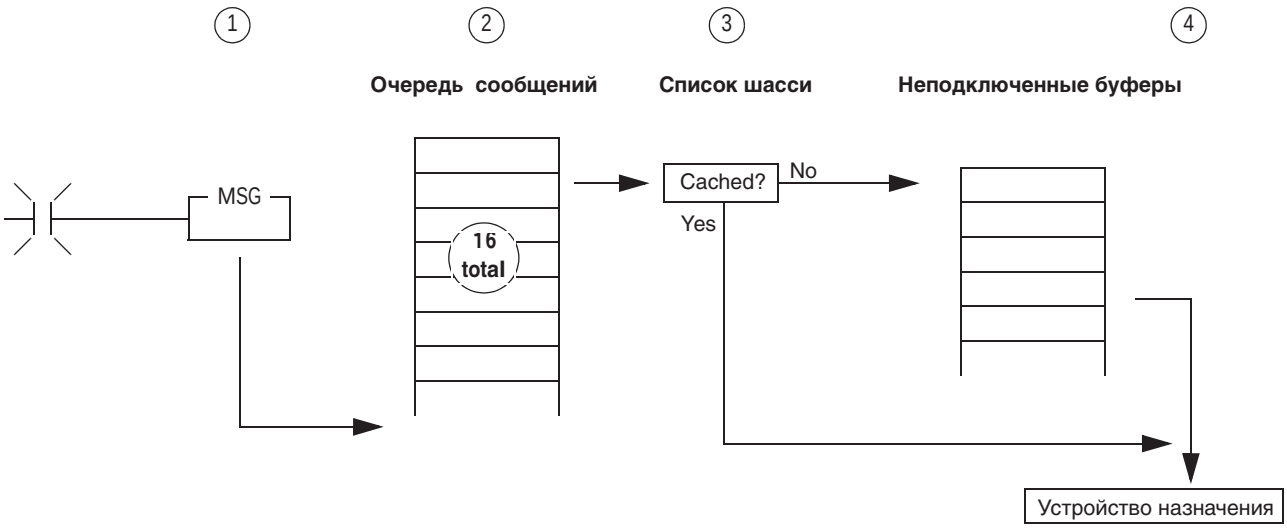
ПРИМЕР

Выполнение инструкции Message (MSG) (Сообщение).

Если *count_send* = 1 и *count_msg.EN* = 0 (инструкция MSG еще не разрешена), то выполните инструкцию MSG, которая пошлет данные другому контроллеру.



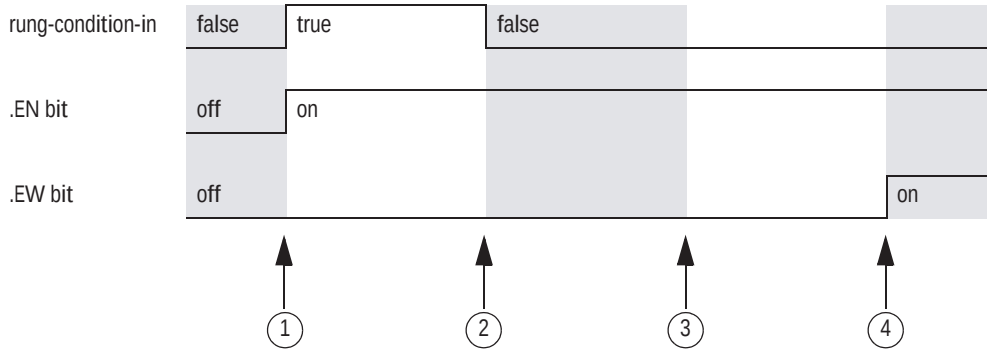
Представленная ниже схема показывает, как контроллер обрабатывает инструкцию Message



Описание:		
1	Контроллер сканирует инструкцию MSG и ее входное условие на значение «истина». Инструкция MSG ставится в очередь сообщений.	
2	Инструкция MSG выходит из очереди сообщений и обрабатывается.	
3	**Если инструкция MSG:**	**То инструкция MSG:**
	Не использует соединение или соединение не было предварительно кэшировано.	Использует неподключенный буфер для установления связи с устройством назначения
	Использует соединение и соединение кэшировано.	Не использует неподключенный буфер
4	Имеет место связь с устройством назначения.	

Очередь сообщений

Очередь сообщений содержит до 16 инструкций, включая те, которые вы сконфигурировали как поблочное чтение, запись. Когда очередь полна, инструкция пытается войти в очередь при каждом последовательном сканировании инструкции, как это показано ниже:



Описание:	
1	Контроллер сканирует инструкцию MSG. Входное условие цепочки имеет значение «истина». Бит EN установлен. Инструкция MSG пытается встать в очередь, но очередь полна (16 инструкций MSG уже разрешены). Бит EW остается очищенным.
2 и 3	Контроллер сканирует инструкцию MSG. Входное условие цепочки имеет значение «ложь». Бит EN остается установленным. Инструкция MSGпытается встать в очередь, но очередь полна. Бит EW остается очищенным.
4	Контроллер сканирует инструкцию MSG. Инструкция MSG пытается встать в очередь. Очередь имеет место для данной инструкции. Бит EW устанавливается.

Список шасси

В зависимости от того, как вы конфигурируете инструкцию MSG, вы можете использовать соединения для отправки и получения данных.

Этот тип сообщения:	И этот способ связи:	Используют соединение:
Запись считывание таблицы CIP	→	x
PLC2, PLC3, PLC5 или SLC (все типы)	CIP CIP с Source ID	
	DH+	x
общий CIP	→	ваша опция(1)
Поблочная запись, считывание	→	x

(1) Вы можете подключать обобщенные сообщения CIP. Но для большинства приложений мы рекомендуем оставлять обобщенные сообщения CIP неподключенными.

Если инструкция MSG использует соединение, у вас имеется опция для того, чтобы оставить это сообщение открытым (кэшировать) или закрыть соединение, когда сообщение передано.

Если вы:	То:
Кэшируете соединение	Соединение остается открытым после выполнения инструкции MSG. Оптимизируется время выполнения. Имеет место открытие соединения всякий раз, как выполняется сообщение, увеличивая время выполнения.
Не кэшируете соединение	Соединение закрывается после выполнения инструкции MSG. Это освобождает данное соединение для других нужд.

Контроллер имеет следующие ограничения на количество соединений, которые вы можете кэшировать:

Если у вас имеется данное программное обеспечение и встроенное ПО версии:	То вы можете кэшировать:
11.x или ранее	• Для поблочной передачи сообщений, до 16 соединений. • Для других типов сообщений, до 16 соединений.
12.x или более поздней	До 32 соединений.

Если в одно и то же устройство направляется несколько сообщений, эти сообщения могут использовать соединение совместно.

Если инструкции MSG	И они:	То:
Для разных устройств	→	Каждая инструкция MSG использует одно соединение.
Для одного устройства	Разрешены одновременно	Каждая инструкция MSG использует одно соединение.
	Не разрешены одновременно	Инструкции MSG используют соединение совместно. (Вместе они учитываются как одно соединение.)

ПРИМЕР

Совместное использование соединения

Если контроллер чередует поблочное чтение сообщения и поблочную запись сообщения для одного модуля, оба сообщения учитываются как одно соединение. Кэширование обоих сообщений учитывается как одно в списке кэширований.

Неподключенные буферы

Контроллер использует неподключенные буферы для установления соединения или обработки сообщений без подключения.

Термин:	Определение:
Неподключенный буфер	Распределение памяти, которую контроллер использует для обработки связей без соединения. Контроллер реализует связь без соединения когда: • Устанавливает соединение с устройством, включая модуль ввода/вывода • Выполняет инструкцию MSG, которая не использует соединение Контроллер может иметь до 10 неподключенных буферов. • Количество по умолчанию – 10. • Для увеличения количества неподключенных буферов выполните инструкцию MSG, которая переконфигурирует количество неподключенных буферов. Обратитесь к разделу «Получение и настройка количества неподключенных буферов» на стр. 10-25. • Каждый неподключенный буфер использует 1.1K памяти. • В ситуации, когда инструкция покидает очередь сообщений, а все неподключенные буферы используются, имеет место ошибка инструкции, и данные не передаются.

Если инструкция MSG использует соединение, эта инструкция использует неподключенный буфер при первом выполнении для установления соединения. Если вы сконфигурировали инструкцию на кэширование соединения, ей более не требуется неподключенный буфер, поскольку соединение установлено.

Указания

При разработке и программировании инструкций MSG пользуйтесь следующими указаниями:

Указание:	Описание:
1. 1 Для каждой инструкции MSG создайте управляющий тег	Каждая инструкция MSG требует свой собственный тег управления • Data type = MESSAGE. • Scope = controller • Этот тег не может быть частью массива или типом данных, определенным пользователем.
2. 2 Сохраняйте данные источника и\или пункта назначения в области контроллера	Инструкция MSG имеет доступ к тегам, расположенным только в папке Controller Tags (Теги контроллера).
3. 3 Если инструкция MSG обращена к устройству, использующему 16-разрядное целое значение, используйте буфер INT в MSG и DINT в проекте.	Если вы отправляете сообщение устройствам, которые используют 16-ти разрядные целые значения, таким как контроллеры PLC-5® или SLC 500™, и оно передает целые значения (не вещественные), используйте буфер INT в этом сообщении и DINT в проекте. Это увеличивает эффективность вашего проекта, поскольку контроллеры Logix5000 работают более эффективно и используют меньше памяти, когда работают с 32 разрядными целыми значениями (DINT). См. раздел “Преобразования типов INT и DINT” на стр. 10-28.
4. 4 Кэшируйте соединенные инструкции MSG, которые выполняются наиболее часто.	Кэшируйте соединения для тех инструкций MSG, которые выполняются наиболее часто, до максимального значения, возможного для версии вашего контроллера. Это оптимизирует время выполнения, потому что контроллер не открывает соединение всякий раз, как выполняется сообщение.
5. 5 Если вы хотите разрешить более 16 инструкций MSG одновременно, используйте определенную стратегию управления.	Если вы разрешаете более 16 инструкций MSG одновременно, некоторые инструкции могут задерживаться с постановкой в очередь. Чтобы гарантировать выполнение каждого сообщения, используйте одну из следующих опций: • Разрешайте каждое сообщение последовательно. • Разрешайте группы сообщений. • Программируйте сообщение для связи с несколькими устройствами. За более подробной информацией обращайтесь к приложению Б. • Программируйте логику на координацию выполнения сообщений. За более подробной информацией обращайтесь к приложению А.
6. 6 Сохраняйте количество несоединенных и некэшированных MSG меньшим, чем количество неподключенных буферов	Контроллер может иметь 10-40 неподключенных буферов. Значение по умолчанию – 10. • Если все неподключенные буферы используются, когда сообщение покидает очередь, имеет место ошибка инструкции и передачи данных не происходит. • Вы можете увеличить количество неподключенных буферов (максимум 40), но продолжайте следовать указанию 5. • Чтобы увеличить количество неподключенных буферов обратитесь на стр. 10-25.

Получение или настройка количества неподключенных буферов

Для определения или изменения количества неподключенных буферов используйте инструкцию MSG.

- Диапазон - от 10 до 40 буферов.
- Количество по умолчанию –10.
- Каждый неподключенный буфер использует 1.1K памяти.

Получение количества неподключенных буферов

Для определения количества неподключенных буферов, которые в данный момент доступны контроллеру, сконфигурируйте инструкцию Message (MSG) следующим образом:

На закладке:	Для этого раздела:	Введите или выберите:	
Configuration (Конфигурация)	Message Type (Тип сообщения)	*CIP Generic*	
	Service Type (Тип службы)	*Custom*	
	Service Code (Код службы)	3	
	Class (Класс)	304	
	Instance (Экземпляр)	1	
	Attribute (Атрибут)	0	
	Source Element (Источник)	*source_array где тип данных = SINT[4]*	
		В этом элементе:	**Введите**
		source_array[0]	1
		source_array[1]	0
		source_array[2]	17
		source_array[3]	0
	Source Length (bytes) (Длина источника)	4 (Запишите 4 SINTs.)	
	Destination (Адрес назначения)	*destination_array where data type = SINT[10] (Сохраните все значения = 0.)*	
		destination_array[6] = текущее количество неподключенных буферов.	
Communication (Связь)	Path (Путь)	*1, slot_number_of_controller*	

Настройка количества неподключенных буферов

В качестве начального значения, настройте количество неподключенных буферов равным количеству неподключенных и некэшированных сообщений разрешенных одновременно плюс примерно 5. 5 дополнительных буферов обеспечивают резерв на случай, если вы недооценили количество разрешенных одновременно сообщений.

Чтобы изменить количество неподключенных буферов контроллера, сконфигурируйте инструкцию Message (MSG) следующим образом:

На закладке:	Для этого раздела:	Введите или выберите:	
Configuration (Конфигурация)	Message Type (Тип сообщения)	*CIP Generic*	
	Service Type (Тип службы)	*Custom*	
	Service Code (Код службы)	4	
	Class (Класс)	304	
	Instance (Экземпляр)	1	
	Attribute (Атрибут)	0	
	Source Element (Источник)	*source_array* где тип данных = SINT[8]	
		В этом элементе:	**Введите**
		source_array[0]	1
		source_array[1]	0
		source_array[2]	17
		source_array[3]	0
		source_array[4]	Количество неподключенных буферов, которое вам требуется.
		source_array[5]	0
		source_array[6]	0
		source_array[7]	0
	Source Length (bytes) (Длина источника)	8 (Запишите 8 SINTs.)	
	Destination (Адрес назначения)	*destination_array* where data type = SINT[6] (Сохраните все значения = 0.)	
Communication (Связь)	Path (Путь)	1, *slot_number_of_controller*	

ПРИМЕР

Настройте количество неподключенных буферов

Если *S:FS* = 1 (первое сканирование), то настройте количество неподключенных буферов для контроллера:

Source_Array[0] = 1

Source_Array[1] = 0

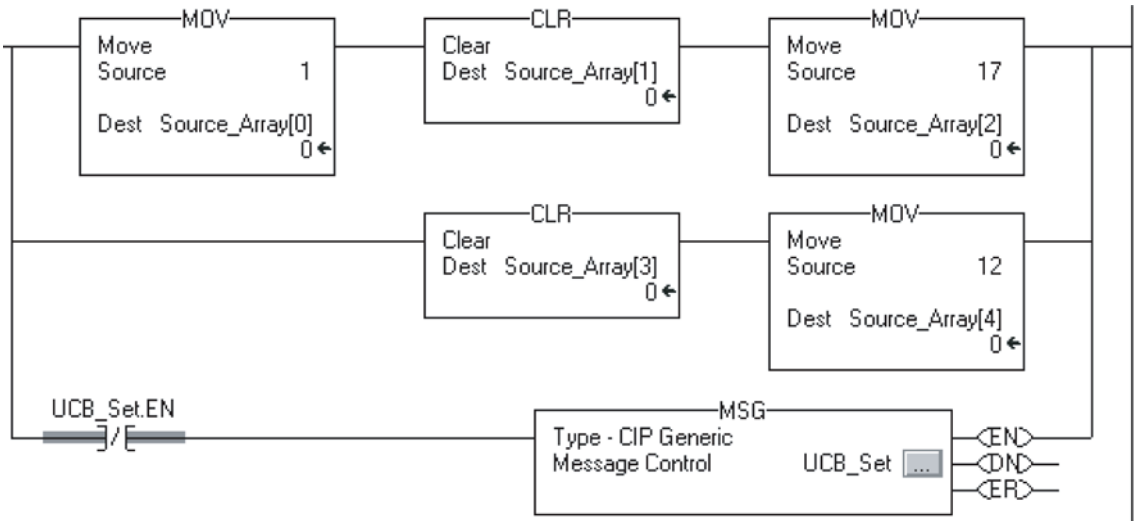
Source_Array[2] = 17

Source_Array[3] = 0

Source_Array[4] = 12 (Количество неподключенных буферов, которое вам требуется. В данном примере вам требуется 12 буферов.)

Если *UCB_Set.EN* = 0 (инструкция MSG еще не разрешена) то:

инструкция MSG устанавливает количество неподключенных буферов = *Source_Array[4]*



где:

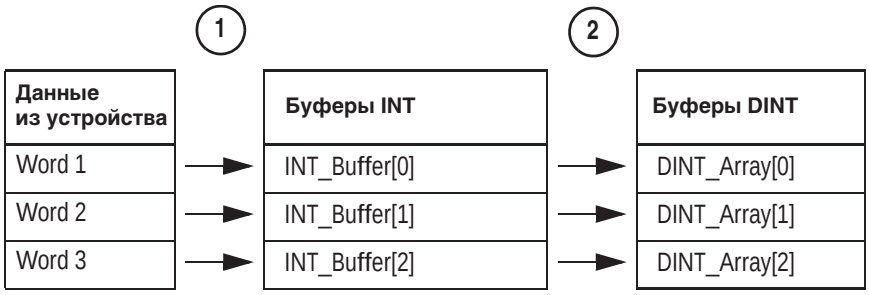
Имя тега:	Тип:	Описание:
UCB_Set	MESSAGE	Управляющий тег для инструкции MSG.
Source_Array	SINT[8]	Значения источника для инструкции MSG, включающее количество неподключенных буферов, которое вам требуется.

Преобразование типов INT и DINT

В контроллере Logix5000 используйте тип данных **DINT** для целых значений данных там, где это только возможно. При использовании 32-х разрядных целых значений контроллеры Logix5000 работают более эффективно и тратят меньше памяти.

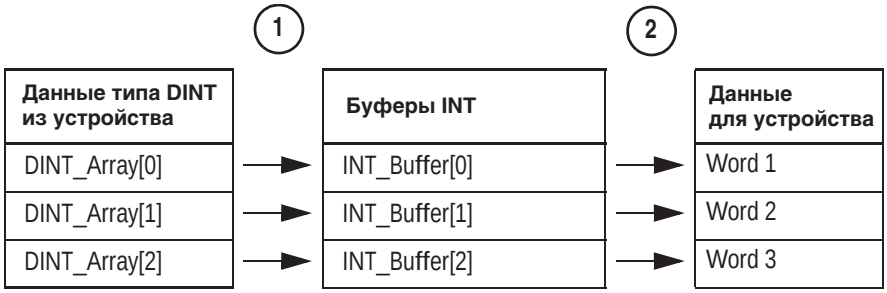
Если ваше сообщение направляется на устройство, использующее 16-ти разрядные целые значения, такие как контроллеры PLC-5® или SLC 500™, и передает целые значения (не вещественные), используйте внутри сообщения буферы INT, и DINT в рамках проекта. Это увеличит производительность вашего проекта.

Считывание 16-разрядных целых значений



1. Инструкция Message (MSG) (Сообщение) считывает 16-разрядные целые значения из устройства и сохраняет их во временном массиве типа INT.
2. Инструкция File Arith/Logical (FAL) преобразует значения типа INT в значения типа DINT для использования другими инструкциями вашего проекта.

Запись 16-разрядных целых значений

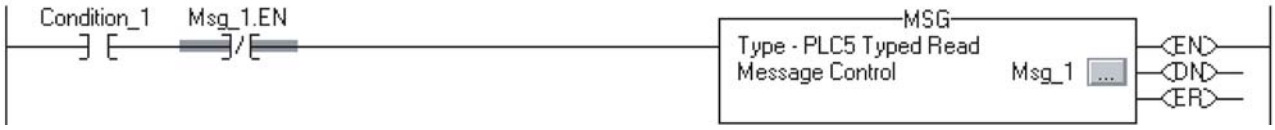


1. Инструкция FAL преобразует данные типа DINT, полученные от контроллера Logix5000, в данные типа INT.
2. Инструкция MSG записывает данные типа INT из временного массива в устройство.

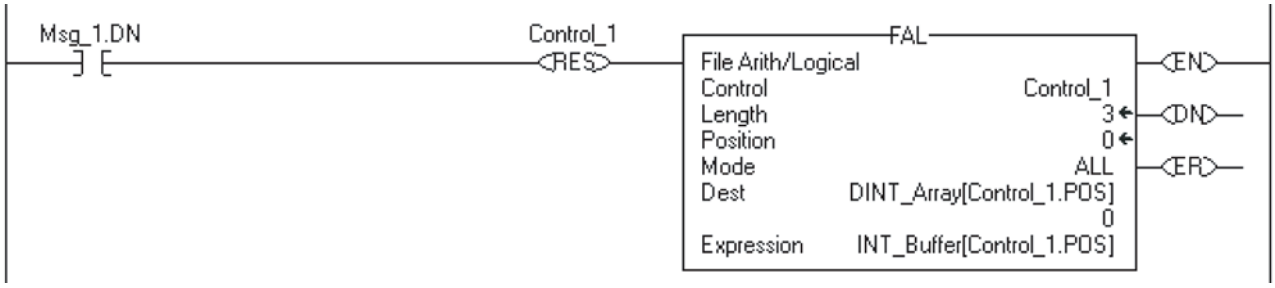
Считывание целых значений из контроллера PLC-5.

ПРИМЕР

Если Condition_1 = 1 And Msg_1.EN = 0 (инструкция MSG еще не разрешена), то:
Считайте 3 целых значения из контроллера PLC-5 и сохраните их в INT_Buffer (3 INTs)



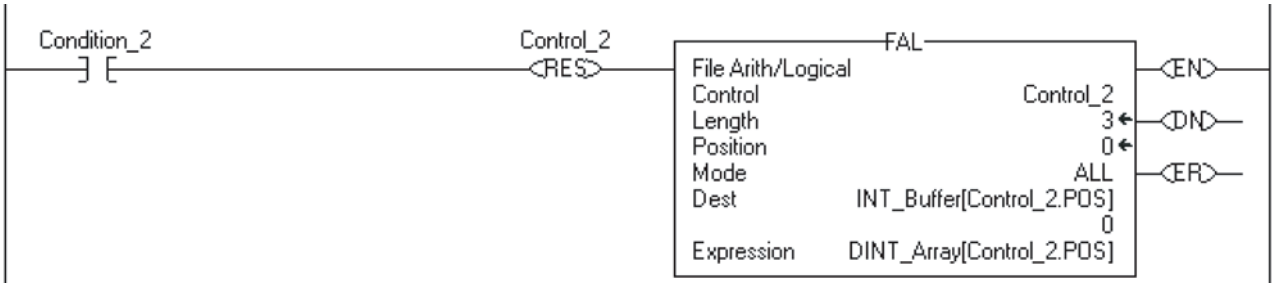
Если Msg_1.DN =1 (инструкция MSG уже считала данные.) то перегрузите инструкцию FAL.
Инструкция FAL устанавливает DINT_Array = INT_Buffer. Это преобразует данные в 32-разрядные целые значения (DINT).



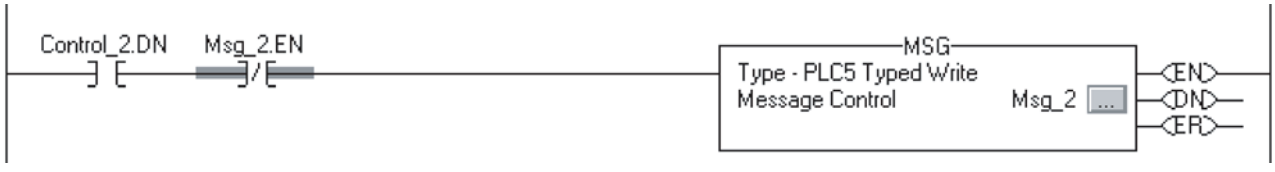
Запишите целые значения в контроллер PLC-5.

ПРИМЕР

Если Condition_2 = 1 то:
Перегрузите инструкцию FAL.
Инструкция FAL устанавливает INT_Buffer = DINT_Array. Это преобразует данные в 16-разрядные целые значения (INT).



Если Control_2.DN = 1 (инструкция FAL уже преобразовала данные типа DINT в тип INT) And Msg_2.EN = 0 (инструкция MSG еще не разрешена) то:
Запишите целые значения в INT_Buffer (3 INT) контроллера PLC-5.



Для заметок:

Создание больших массивов

Когда использовать данную процедуру

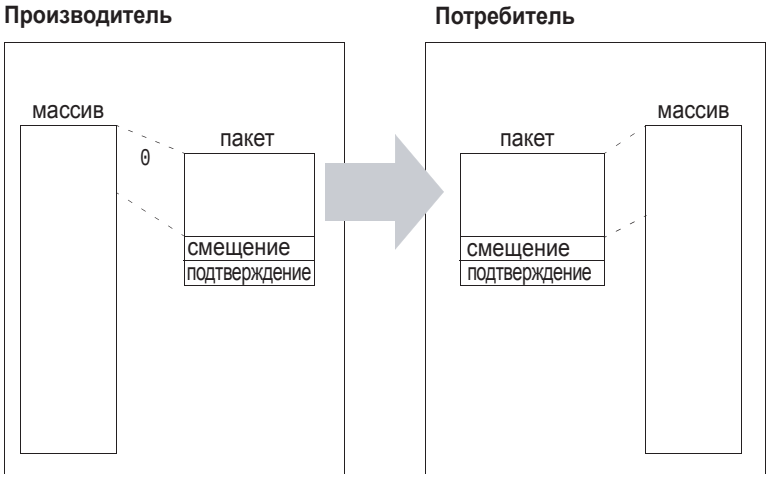
Контроллеры Logix5000 могут посылать 500 байт данных через одно запланированное соединение. Это соответствует массиву из 125 элементов типа DINT или REAL. Для передачи более чем 125 элементов DINT или REAL используйте пакет данных, создаваемый с использованием производящего/потребляющего тега из 125 элементов. Вы можете использовать этот пакет для кусочной пересылки массива в другой контроллер.

Когда вы посылаете большой массив данных маленькими пакетами, вы должны удостовериться, что передача пакета завершена до перемещения данных в массив назначения по следующим соображениям:

- Произведенные данные посылаются через системную плату ControlLogix сегментами по 50 байт.
- Передача данных происходит асинхронно по отношению к сканированию программы.

Алгоритм, включенный в этот раздел, использует подтверждающее слово, чтобы убедиться, что каждый пакет содержит новые данные прежде, чем данные переместятся в массив-адресат. Этот алгоритм также использует величину смещения, чтобы указать начальный элемент пакета в массиве.

Из-за элементов подтверждения и смещения каждый пакет несет 123 элемента данных массива, как это изображено ниже:



Кроме того, массив должен содержать дополнительные 122 элемента. Другими словами, он должен быть на 122 элемента больше, чем наибольшее число элементов, которые вы хотите передать:

- Эти элементы служат в качестве буфера.
- Поскольку каждый пакет содержит одинаковое число элементов, то буфер предохраняет контроллер от копирования за границами массива.
- Без буфера это должно иметь место, если последний пакет содержит меньше 123 фактических элементов данных.

Создание большого массива

1. Откройте проект RSLogix 5000, который будет создавать массив.
2. В области папки тэгов контроллера Controller Tags создайте следующие тэги:

P	Имя тега:	Тип:
	*array*_ack	DINT[2]
x	*array*_packet	DINT[125]

где:

array – имя данных, которые вы будете посылать.

3. Преобразуйте *array_ack* в потребляемый тэг:

Для:	Задайте:
Controller	Имя контроллера, получающего пакет
Remote Tag Name	*array_ack* для этих совместных данных оба контроллера используют одно и то же имя

Смотрbте: «Получение данных, созданных другим контроллером » на странице 10-15.

4. Создайте следующие тэги в папке Controller Tags или в папке тэгов программы, которая будет содержать алгоритм для передачи:

Имя тега:	Тип:
array	DINT [x], где x равно числу элементов для пересылки плюс 122 элемента
*array*_offset	DINT
*array*_size	DINT
*array*_transfer_time	DINT
*array*_transfer_time_max	DINT
*array*_transfer_timer	TIMER

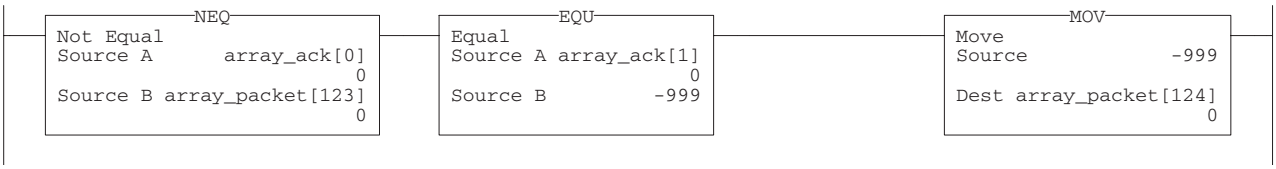
где:

array –имя данных, которые вы будете посылать.

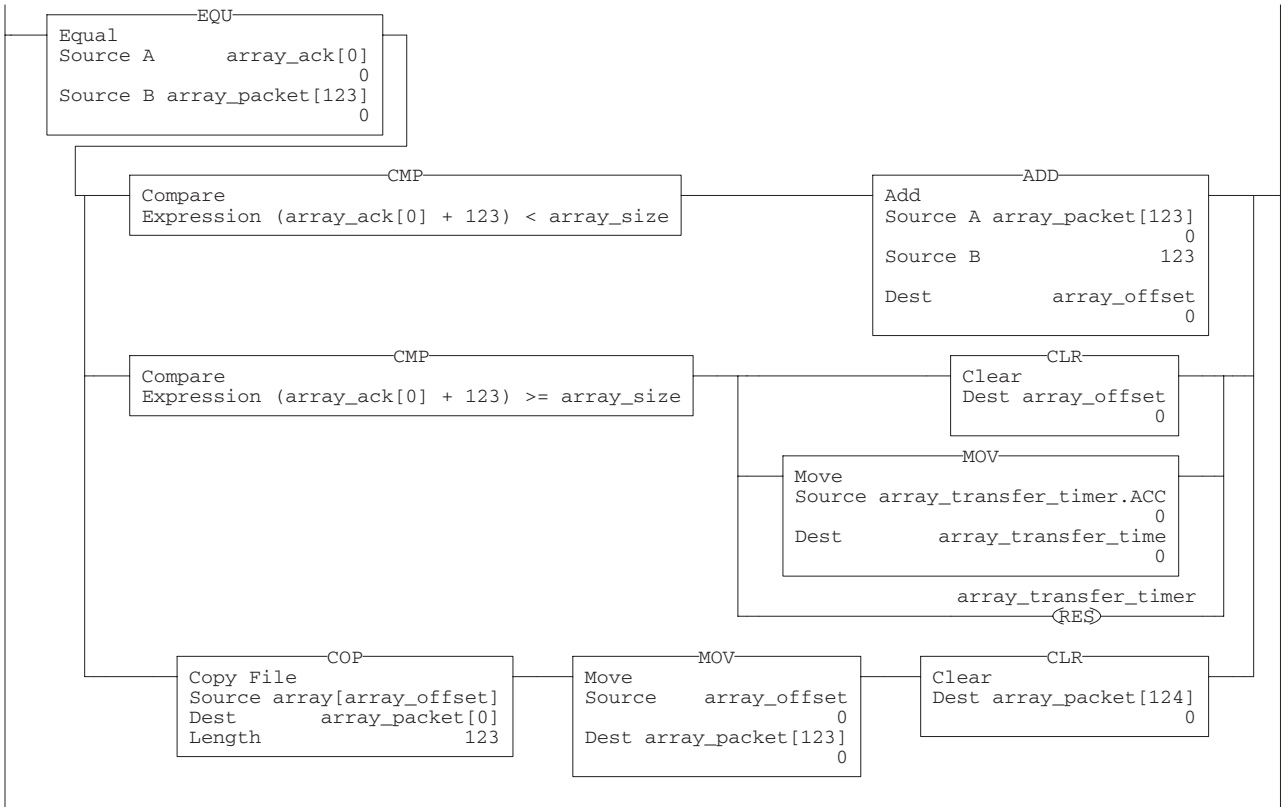
5. В *array_size* введите число элементов реальных данных. (Значение *x* из шага 4 минус 122 элемента буфера.)
6. Создайте или откройте процедуру для логики которая будет создавать пакеты данных.
7. Введите следующий алгоритм:

Сколько времени нужно для передачи всего массива

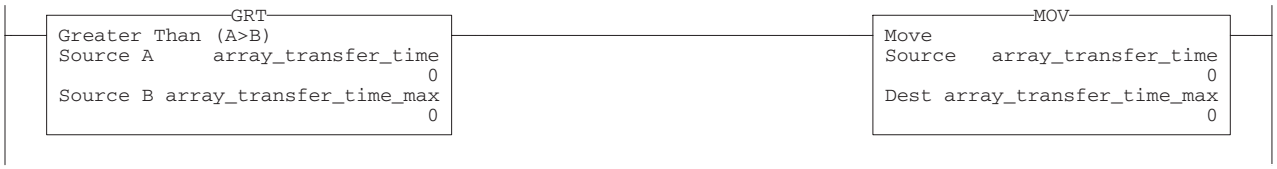
Когда значение смещения в *array_ack [0]* не равно текущему значению смещения, а *array_ack [1]* равен –999 и потребитель уже начал получать новый пакет, то цепь перемещает -999 в последний элемент пакета. Потребитель ожидает пока он не получит значение -999 прежде, чем скопирует пакет в массив. Это гарантирует, что потребитель получил новые данные.



Если значение смещения в *array_ack [0]* равно текущему значению смещения и потребитель уже скопировал пакет в массив, то цепочка проверяет наличие данных для перемещения. Если значение смещения плюс 123 меньше, чем размер массива и есть еще данные для перемещения, то цепь увеличивает смещение на 123. В противном случае, если нет больше данных для перемещения, цепь сбрасывает значение смещения, регистрирует время передачи и сбрасывает таймер. В любом случае, цепь использует новое значение смещения для создания нового пакета данных, добавляет новое значение смещения в пакет и очищает элемент подтверждения пакета (*packet [124]*).



Если текущее время передачи больше чем максимальное время передачи, то обновляется максимальное время передачи. Это сохраняет запись наибольшего времени передачи данных.



42355

8. Откройте проект RSLogix 5000, который будет потреблять массив.

9. В области контроллерных тэгов создайте следующие тэги:

P	Имя тэга	Тип
x	*array*_ack	DINT [2]
	*array*_packet	DINT [125]

где:

array – имя данных, которые вы будете посылать. Используйте тоже имя, что и в производящем массив контроллере (шаг 2).

10. Преобразуйте *array_packet* в потребленный тэг:

Для:	Задать:
контроллера	имя контроллера, который будет посылать пакеты
имени удаленного тега	*array_packet* Оба контроллера используют одинаковое имя для этих совместных данных.

Смотри «Использование тегов, созданных другим контроллером » на странице 10-15.

11. В папке Controller Tags или в папке тэгов программы, которая будет содержать логику для передачи, создайте следующие тэги:

Имя тэга	Тип
array	DINT [*x*],где *x* равно числу элементов для пересылки плюс 122 элемента
*array*_offset	DINT

где:

array –имя данных, которые вы будете посылать.

12. Создайте или откройте процедуру для алгоритма, который будет перемещать данные из пакетов в массив назначения.

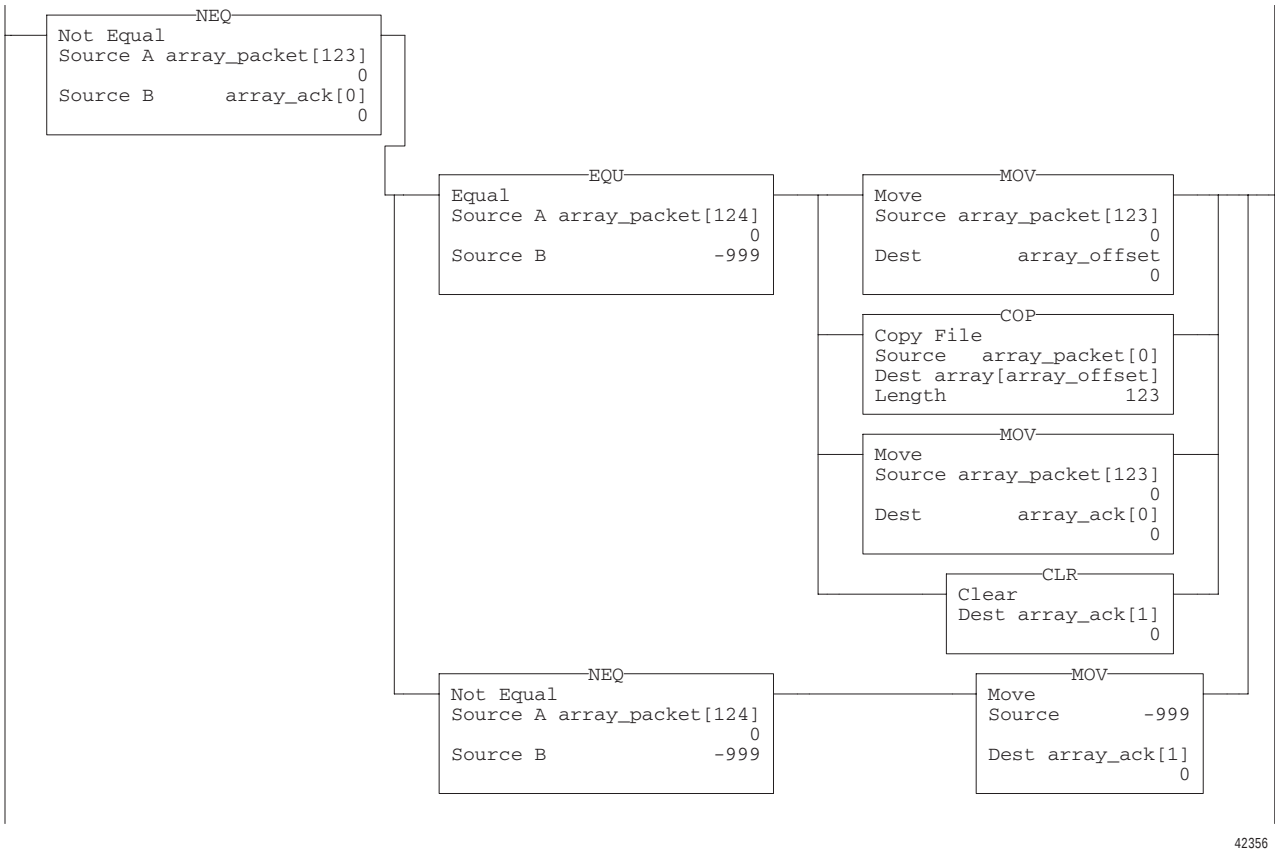
13.Введите следующую логику:

Если значение смещения в *array_packet [123]* отлично от значения смещения в *array_ack [0]*, контроллер уже начал получать новый пакет данных; то цепь проверяет наличие значения -999 в последнем элементе пакета.

Если последний элемент пакета равен -999, контроллер уже получил целый пакет новых данных и начинал операцию копирования:

- Значение смещения перемещается из пакета в *array_offset* .
- Инструкция COP копирует данные из пакета в массив назначения, начиная со значения смещения.
- Значение смещения перемещается в *array_ack [0]*, что показывает, что копирование завершено.
- *Array_ack [1]* устанавливается в ноль и ожидает сигнализации о прибытия нового пакета.

Если последний элемент пакета не равен -999 и передача пакета в контроллер может быть не завершенна, то -999 перемещается в *array_ack [1].*Это сигнализирует производителю о возврате значения -999 в последний элемент пакета для проверки передачи пакета.



Передача большого массива в виде меньших пакетов повышает производительность системы по сравнению с другими способами передачи данных:

- Используется меньше соединений, чем если бы вы разбили данные на множество массивов и посылали каждый, как тег. Например, массив из 5000 элементов потребует 40 связей (5000/125=40), используемых отдельными массивами.
- Достигается более быстрое время передачи, чем если бы вы использовали инструкцию сообщения (message), чтобы послать целый массив.
  - Сообщения незапланированы и выполняются только в течение «системного» (system overhead) этапа работы Logix5550. Следовательно, сообщения могут использовать довольно много времени, для выполнения передачи данных.
  - Вы можете уменьшить время передачи, увеличивая отрезок времени системный работы, но это уменьшит производительность непрерывной задачи.

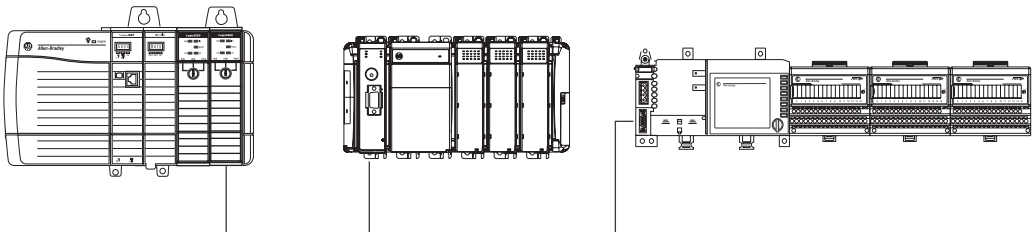
Для заметок:

Связь с устройствами ASCII

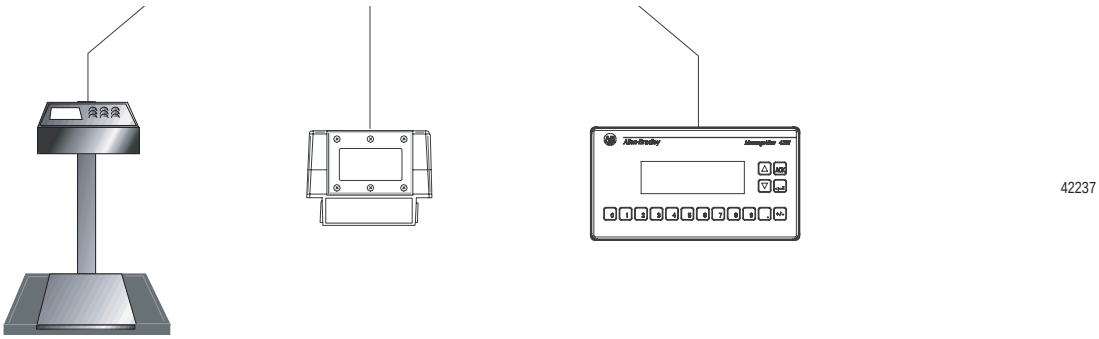
Когда используется данная процедура

Используйте данную процедуру для обмена ASCII данными с каким-либо устройством при помощи последовательного порта. Например, вы можете использовать последовательный порт для:

- Считывания символов ASCII с модуля весов или считывателя штрихового кода,
- Отправки и получения сообщений от триггерных устройств, таких как терминал MessageView (Просмотр сообщений).



связь от последовательного порта контроллера к устройствам ASCII



42237

Как пользоваться данной процедурой

Перед использованием данной процедуры:

- Сконфигурируйте устройства ASCII под ваше приложение

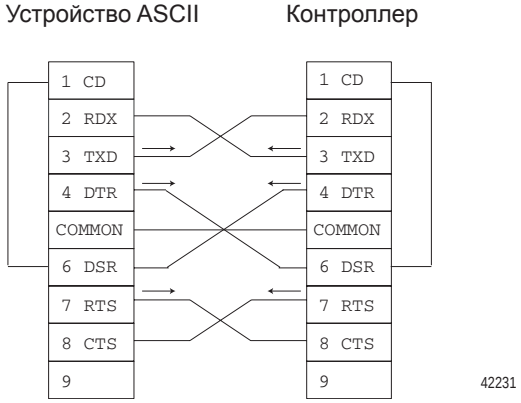
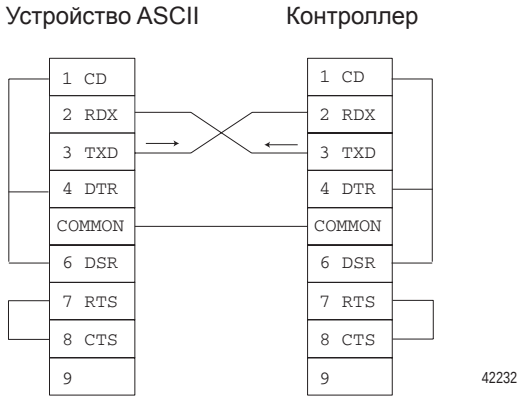
Чтобы выполнить эту процедуру сделайте следующее:

- Подключите устройство ASCII
- Сконфигурируйте последовательный порт
- Сконфигурируйте протокол пользователя
- Создайте типы строковых данных
- Считайте символы с устройства.

- Отправьте символы на устройство.

Подключение устройства ASCII

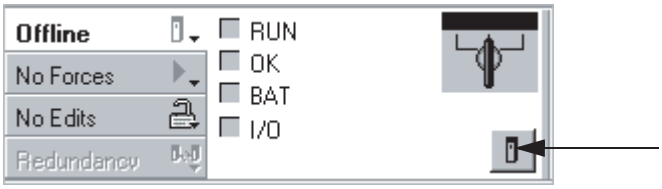
1. Для последовательного порта устройства ASCII задайте, какие контакты отправляют сигналы, а какие получают.
2. Соедините контакты отправки сигнала с соответствующими контактами получения и установите перемычки:

Если соединение	Тогда подключите коннекторы следующим образом
С квитированием	
Без квитирования	

3. Подключите экран кабеля к обоим коннекторам.
4. Подключите кабель к контроллеру и устройству ASCII.

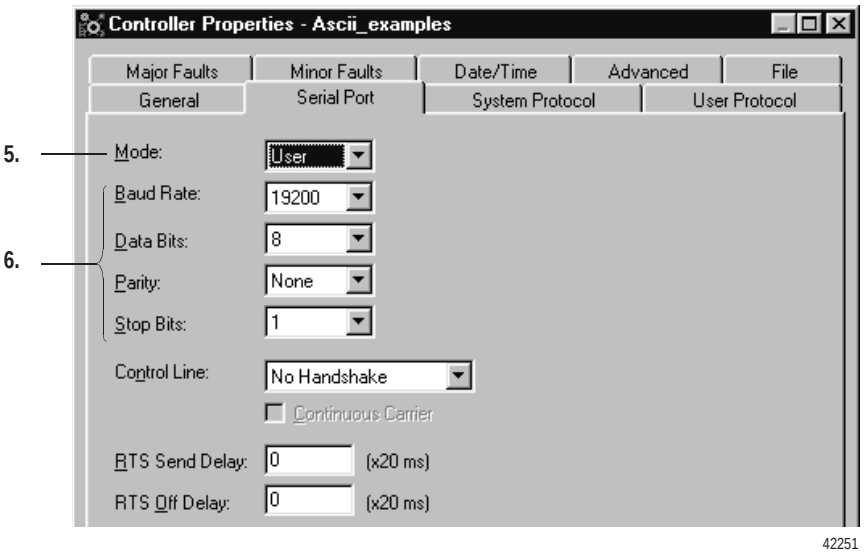
Конфигурирование последовательного порта

1. Задайте следующие настройки связи для устройства ASCII:
   a. скорость передачи в бодах;
   b. биты данных;
   c. четность;
   d. стоповые биты.
2. Откройте проект RSLogix5000.



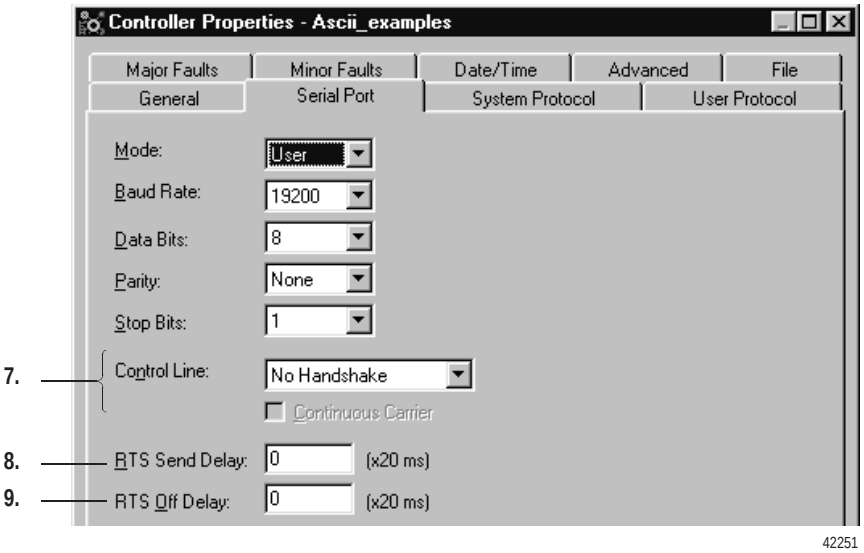
42627

3. На панели инструментов Online щелкните на кнопке контроллера
4. Щелкните на закладке *Serial Port* (Последовательный порт).



42251

5. Выбирите *User* (Пользователь).
6. Выбирите настройки для устройства ASCII из шага 1.



7. Выбирите опцию Control Line (Служебная строка)

Если:	**И:**	**И:**	**Выберите:**	**Затем:**
вы не используете модем	→		*No Handshaking*	Перейти к шагу 10.
вы используете модем	оба модема при соединении точка-точка имеют полнодуплексный режим	→	*Full Duplex*	
	Ведущий модем имеет полнодуплексный, а подчиненный модем полудуплексный режим	Ведущий контроллер	*Full Duplex*	
		Подчиненный контроллер	*Half Duplex*	Выберите опцию *Continuous Carrier*.
	все модемы в системе имеютполудуплексный режим	→	*Half Duplex*	Сбросьте опцию *Continuous Carrier* (по умолчанию).

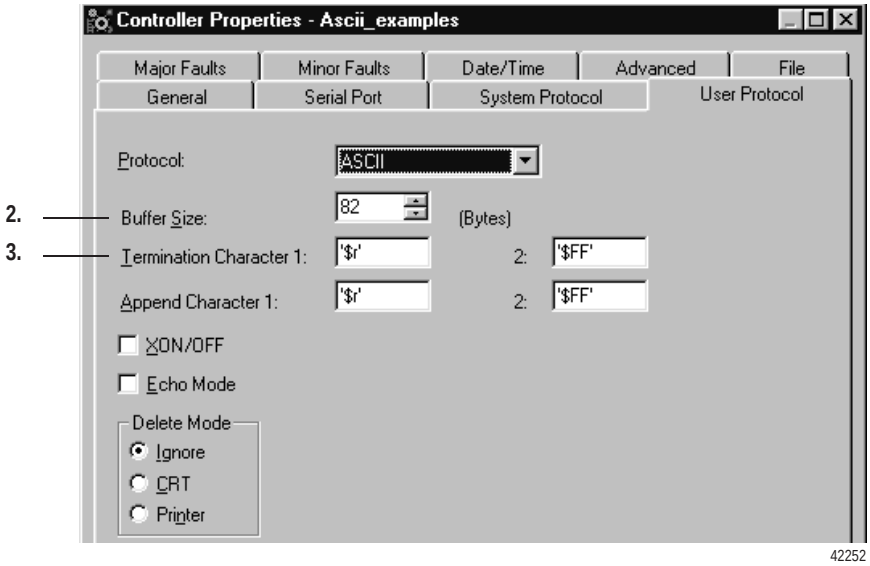
8. Введите величину задержки (20 мс на единицу) между временем включения сигнала RTS (ВКЛ) и временем отправки данных. Например, величина 4 соответствует 80 мс задержки.

9. Введите величину задержки (20 мс на единицу) между временем отправки последнего символа и временем отключения сигнала RTS (нижнее).

10.Щелкните на *Apply*.

Конфигурирование протокола пользователя

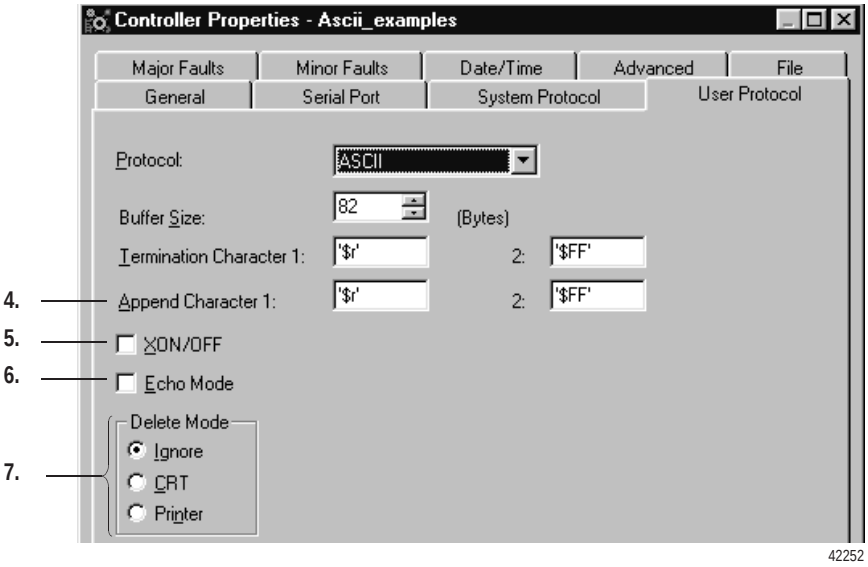
1. Щелкните на закладке *User Protocol* (Протокол пользователя).



2. Выберите или введите число, которое больше или равно наибольшему числу символов в передаче. (Двойное количество символов является хорошим выбором.)

3. Если вы используете инструкции ABL или ARL, введите символ, который помечает конец данных. Символы ASCII кодов смотрите в конце этого руководства.

Если устройство посылает:	**То:**	**Примечание:**
один символ окончания	A. В текстовом окне Termination Character 1 введите шестнадцатеричный код ASCII для первого символа. B. В текстовом окне Termination Character 2 введите $FF	Для печатных символов, таких как 1 или A, введите этот символ.
два символа окончания	В текстовых окнах Termination Character 1 и Termination Character 2 введите шестнадцатеричный ASCII код для каждого символа.	



4. Если вы используете инструкцию AWA, введите символ(ы), который обозначает добавление данных. Символы ASCII кодов смотрите в конце этого руководства.

Для добавления:	То:	Примечание:
одного символа	A. В текстовом окне Append Character 1 введите шестнадцатеричный ASCII код для первого символа. B. В текстовом окне Append Character 2 введите $FF	Для печатных символов, таких как 1 или A, введите этот символ.
два символа окончания	В текстовых окнах Append Character 1 и Append Character 2 введите шестнадцатиричный ASCII код для каждого символа.	

5. Если устройство ASCII конфигурируется на управление потока XON/XOFF, установите флажок *XON/XOFF*.

6. Если устройство ASCII является устройством CRT или сконфигурировано для полудуплексной передачи, установите флажок *Echo Mode*.

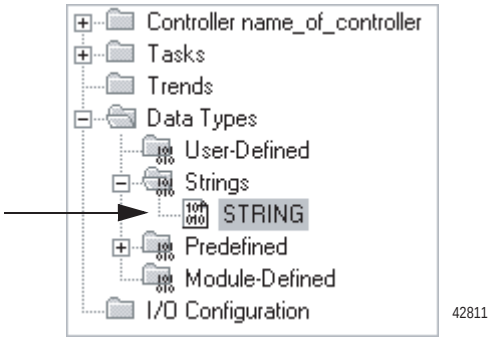
7. Выберите Delete Mode (Режим удаления):

Если устройство ASCII является:	**Выберите:**	**Примечание:**
CRT	*CRT*	• Символ DEL ($7F) и символ, который предшествует символу DEL, не будут отправлены в устройство назначения. • Если выбран режим ECHO и инструкция ASCII считывает символ DEL, эхо возвращает три символа: BACKSPACE SPACE BACKSPACE ($08 $20 $08).
принтер	*Printer*	• Символ DEL ($7F) и символ, который предшествует символу DEL, не будут отправлены в устройство назначения. • Если выбран режим ECHO и инструкция ASCII читает символ DEL, эхо возвращает два символа: /($2F) за которыми следует символ, который был удален.
ни одно из предыдущих	*Ignore*	Символ DEL ($7F) отрабатывается как любой другой символ.

8. Щелкните на OK.

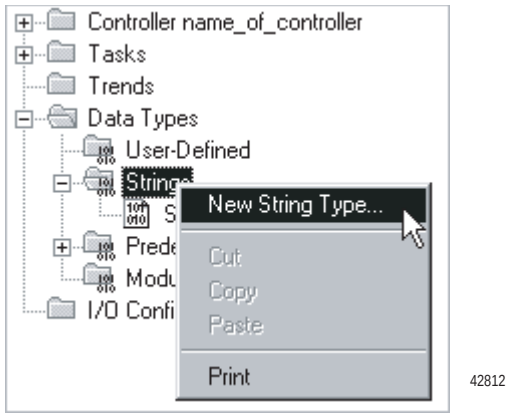
Создание типов строковых данных

Вы сохраняете символы ASCII в тегах, которые используют строковый тип данных.



Вы можете использовать строковый тип данных (STRING) по умолчанию. Он позволяет хранить до 82 символов.

или



Вы можете создать новый строковый тип данных, который будет хранить столько символов, сколько вы зададите.

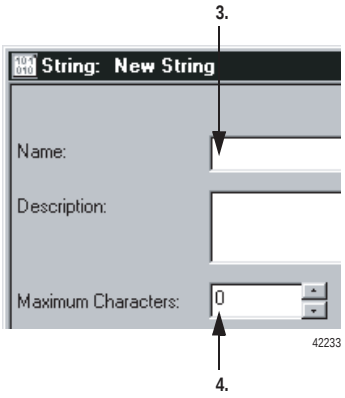
ВАЖНО

Будьте внимательны, когда создаете строковый тип данных. Если позднее вы решите изменить размер строкового типа данных, вы можете потерять данные в тегах, которые в этот момент используют этот тип данных.

Если вы:	То:
Уменьшаете размер строкового типа данных	• Данные отбрасываются • LEN не изменяется
Увеличиваете размер строкового типа данных	Данные и LEN сбрасываются на 0

1. Вы хотите создать новый строковый тип данных?

Если:	То:
нет	См. раздел «Считывание символов из устройства» на стр. 12-9.
да	Переходите к шагу 2.



2. В организаторе контроллера щелкните правой клавишей мыши на *Strings(Строки)* и выберите *New String Type...* (*Новый тип строковых данных*).

3. Введите имя для этого типа данных.

4. Введите максимальное количество символов, которые будет хранить этот тип данных.

5. Выберите *OK*.

Считывание данных из устройства

Пользуйтесь следующим общим правилом: перед чтением буфера используйте инструкцию ACB или ABL для проверки того, что буфер содержит требуемые символы:

- Инструкция ARD или ARL продолжает считывать буфер, пока инструкция не считает требуемые символы.
- Когда инструкция ARD или ARL осуществляет чтение буфера, никакие другие инструкции последовательного порта ASCII (ASCII Serial Port) не выполняются, за исключением ACL.
- Проверка того, что буфер содержит требуемые символы, предотвратит возможность задержки инструкцией ARD или ARL выполнения других инструкций последовательного порта ASCII пока входное устройство посылает свои данные.

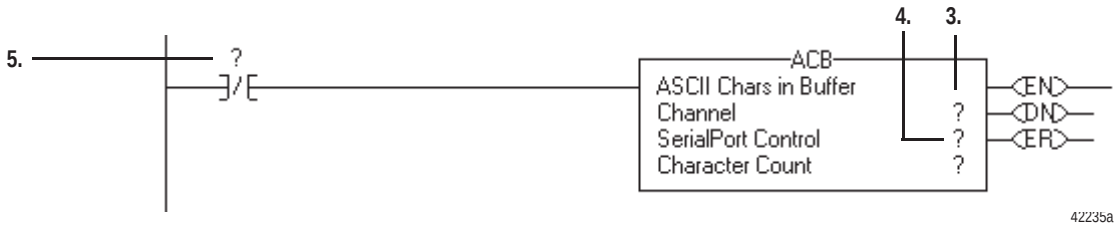
Дополнительную информацию о инструкциях последовательного порта ASCII вы надете в справочном руководстве "*Общие инструкции контроллера Logix5000" (Controllers General Instruction Set Reference Manual),* документ 1756-RM003.

ВАЖНО Если вы не знакомы с программированием на языке релейной логики в проекте RSLogix 5000, то обратитесь к разделу "Программирование на языке релейной логики" на стр. 8-1.

1. Из устройства какого типа вы производите считывание?

Если устройство:	То:
Считыватель штрихового кода	Перейдите к шагу 2
Весы, которые передают фиксированное количество символов	
Терминал сообщений или дисплей	Перейдите к шагу 14
Весы, которые передают переменное количество символов	

2. Введите следующую цепочку:

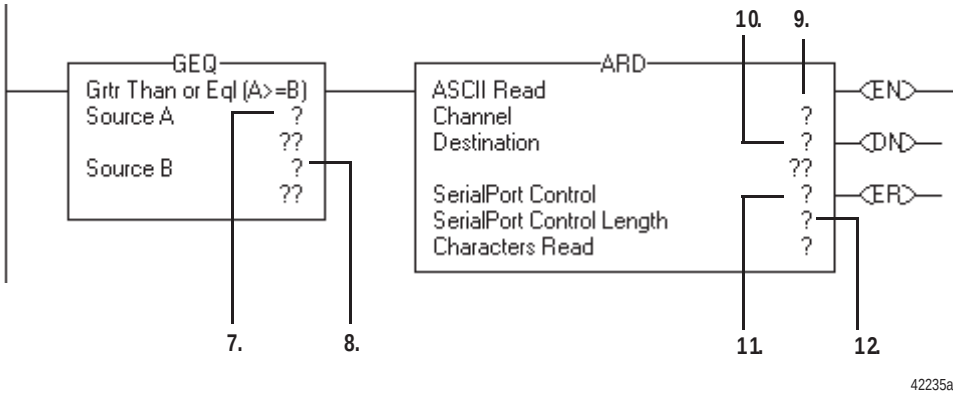


3. Введите 0. (Последовательный порт – это канал 0.)

4. Введите имя тега для инструкции ACB и определите тип данных как SERIAL_PORT_CONTROL.

5. Введите бит EN тега AC (тег из шага 4.)

6. Введите следующую цепочку:



7. Введите член POS тега ACB. (Тег из шага 4.)

8. Введите количество символов в данных.

9. Введите 0.

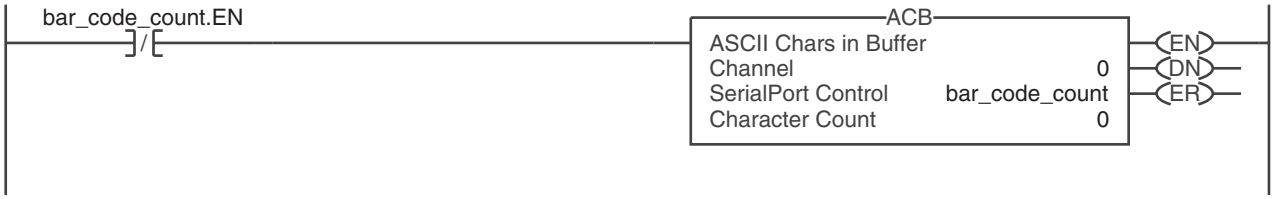
10. Введите имя тега для хранения символов ASCII. Определите тип данных как **строковый**.

11. Введите имя тега для инструкции ARD и определите тип данных как SERIAL_PORT_CONTROL.

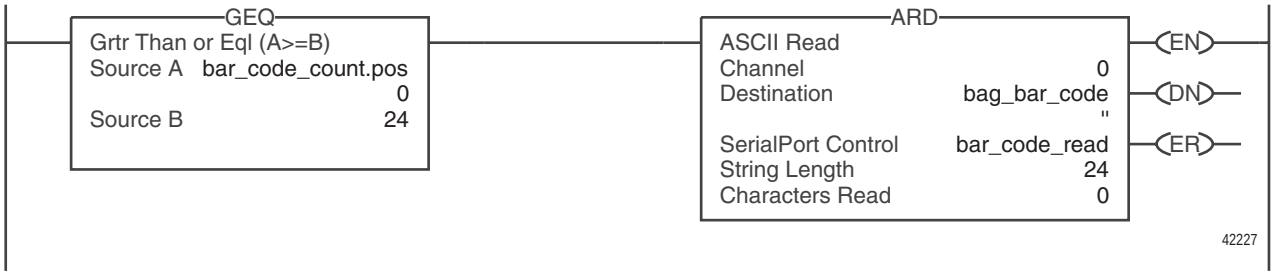
12. Введите количество символов в данных.

ПРИМЕР

Считыватель штрихового кода отсылает штриховые коды в последовательный порт (канал 0) контроллера. Каждый штриховой код содержит 24 символа. Для того чтобы определить, когда контроллер получил штриховой код, инструкция ACB продолжает подсчитывать символы в буфере.



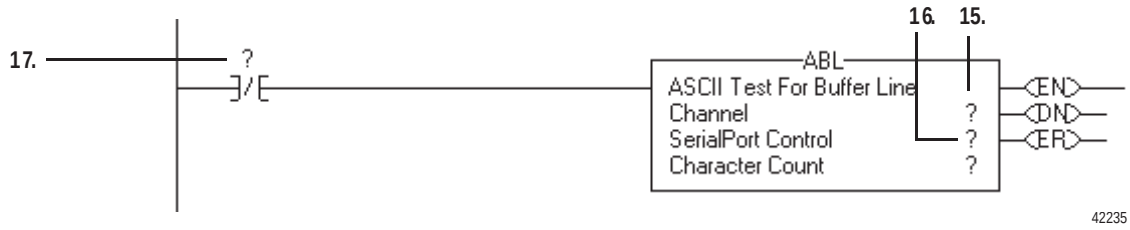
Если буфер содержит 24 символа, это означает, что контроллер получил штриховой код. Инструкция ARD перемещает этот штриховой код в тег *bag_bar_code*.



13. Вы хотите отправить данные в устройство?

Если:	То:
да	Перейдите к разделу “Отправка символов в устройство” на стр. 12-14.
нет	Остановитесь. Вы завершили эту процедуру. Описание использования данных вы найдете в разделе “Обработка символов ASCII” на стр. 13-1.

14.Введите следующую цепочку:



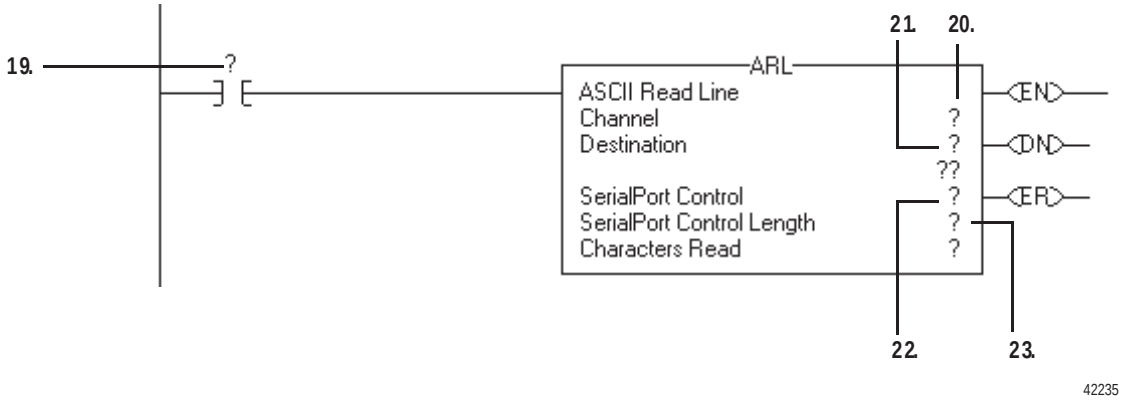
15 .

15.Введите 0.

16.Введите имя тега для инструкции ABL и определите тип данных как SERIAL_PORT_CONTROL.

17.Введите бит EN тега ABL. (Тег из шага 16.)

18.Введите следующую цепочку:



19.Введите бит FD тега ABL. (Тег из шага 16.)

20.Введите 0.

21.Введите имя тега для сохранения символов ASCII. Определите тип данных как **строковый**.

22.Введите имя тега для инструкции ARL и определите тип данных как SERIAL_PORT_CONTROL.

23.Введите 0.

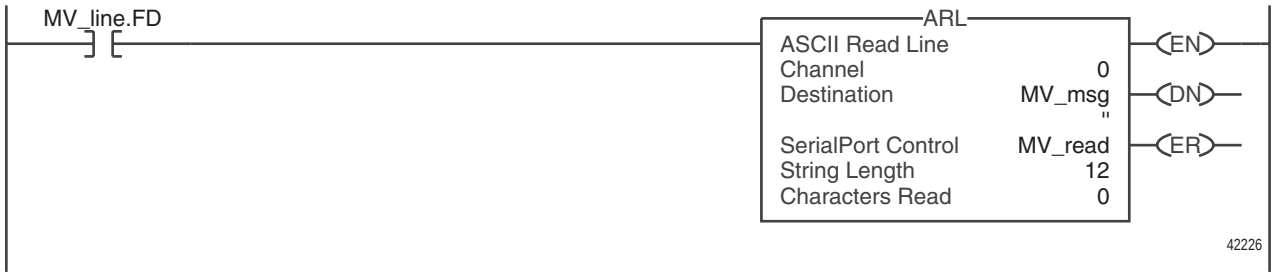
Это позволит инструкции установить SerialPort Control Length (Контрольная длина для последовательного порта) равной размеру Destination (Пункт назначения).

ПРИМЕР

Непрерывно проверяйте буфер на наличие сообщения с терминала MessageView.

- Поскольку каждое сообщение завершается символом возврата каретки ($0D), возврат каретки конфигурируется как символ завершения в закладке User Protocol (Протокол пользователя0 диалогового окна Controller Properties (Свойства контроллера).
- Когда ABL находит возврат каретки, она устанавливает бит FD.

Когда инструкция ABL находит возврат каретки (устанавливается MV_line.FD), контроллер удаляет символы из буфера до и включая символ возврата каретки и размещает их в теге *MV_msg*.



24. Вы хотите отправить данные в устройство?

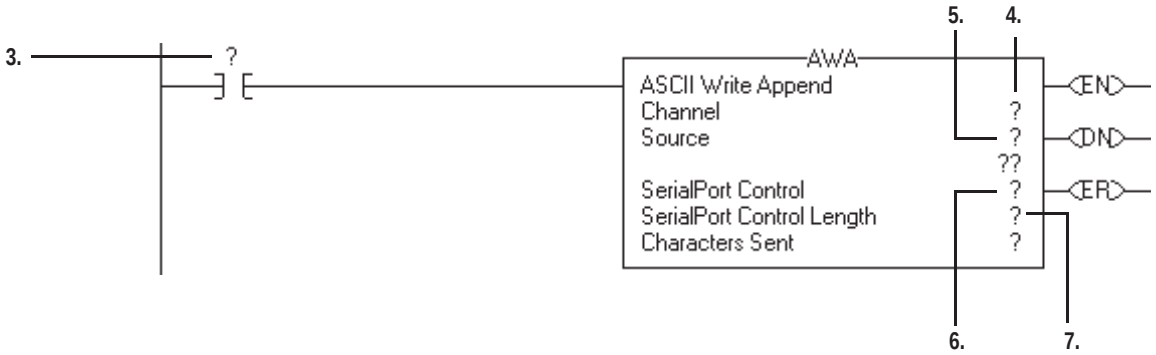
Если:	То:
да	Перейдите к разделу “Отправка символов в устройство” на стр. 12-14.
нет	Остановитесь. Вы завершили эту процедуру. Описание использования данных вы найдете в разделе “Обработка символов ASCII” на стр. 13-1.

Отправка символов в устройство

1. Определитесь, с чего начать:

Если вы:	И вы:	То:
Всегда отправляете одно и то же количество символов	Хотите автоматически добавлять один или два символа к концу данных	Перейдите к шагу 2
	Не хотите добавлять символы	Перейдите к шагу 9
Отправляете разное количество символов	Хотите автоматически добавлять один или два символа к концу данных	Перейдите к шагу 16
	Не хотите добавлять символы	Перейдите к шагу 24

2. Введите следующую цепочку:



42236a

3. Введите входное условие (условия), которое задаст момент отправки:

- Вы можете использовать любой тип входной инструкции.
- Эта инструкция должна изменять значение с “ложь” на “истина” каждый раз, когда должны быть отправлены данные.

4. Введите 0.

5. Введите имя тега, в котором сохраняются символы ASCII. Определите тип данных как **строковый**.

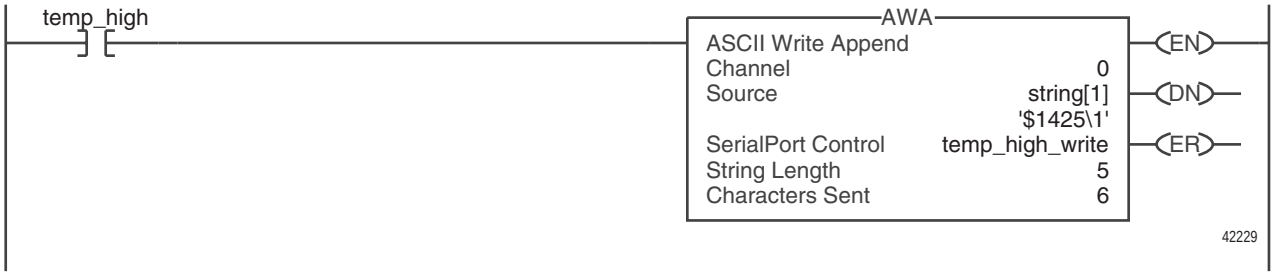
6. Введите имя тега для инструкции AWA и определите тип данных как SERIAL_PORT_CONTROL.

7. Введите количество символов для отправки. Не включайте символы, добавляемые инструкцией.

ПРИМЕР

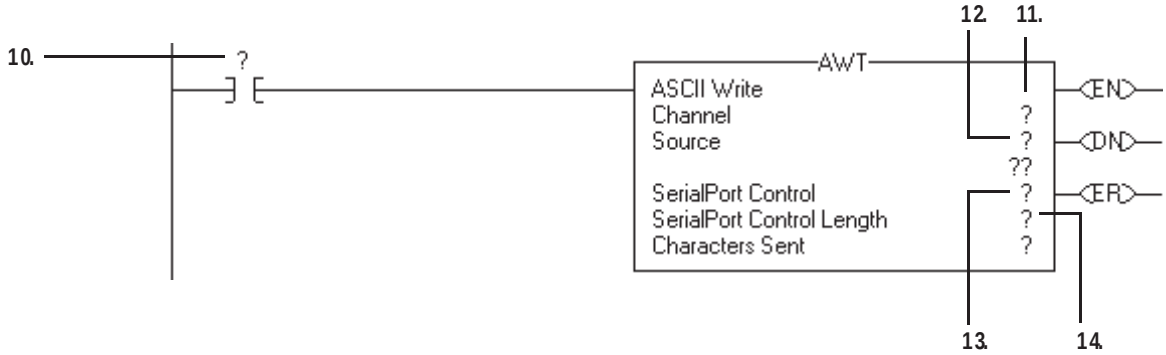
Когда температура превышает верхний предел (установлен *temp_high*), инструкция AWA отправляет 5 символов из тега *string[1]* в терминал MessageView.

- *$14* засчитывается за 1 символ. Это шестнадцатеричный код для символа Ctrl-T.
- Инструкция также посылает (добавляет) символы, заданные в протоколе пользователя. В этом примере инструкция AWA посылает возврат каретки ($0D), который помечает конец данного сообщения.



8. Переходите к разделу «Ввод символов ASCII» на стр. 12-21.

9. Введите следующую цепочку:



10. Введите входное условие (условия), которое задаст момент отправки:

- Вы можете использовать любой тип входной инструкции.
- Эта инструкция должна изменять значение с “ложь” на “истина” каждый раз, когда должны быть отправлены данные.

11. Введите 0.

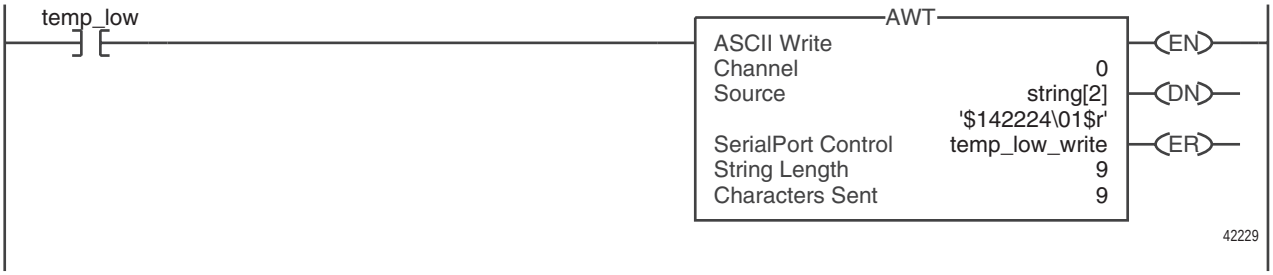
12. Введите имя тега, в котором сохраняются символы ASCII. Определите тип данных как **строковый**.

13. Введите имя тега для инструкции AWT и определите тип данных как SERIAL_PORT_CONTROL.

14. Введите количество символов для отправки.

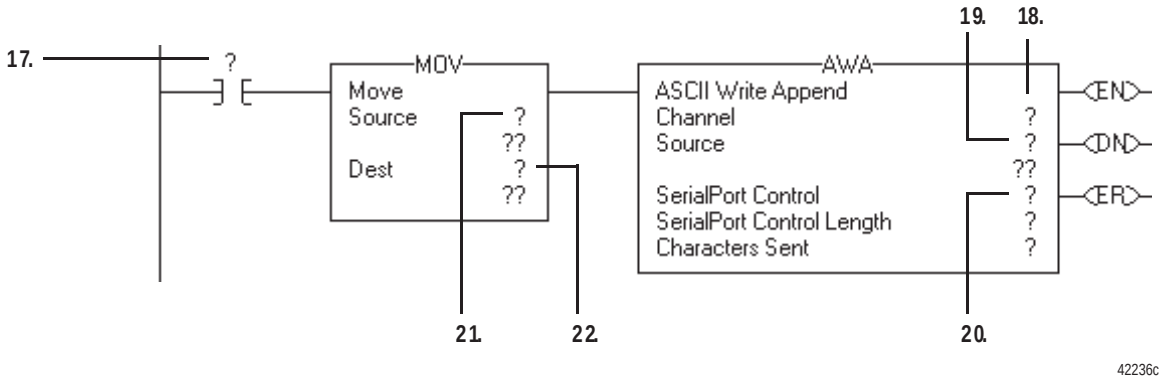
ПРИМЕР

Когда температура достигает нижнего предела (установлен *temp_low*), инструкция AWT отправляет 9 символов из тега *string[2]* в терминал MessageView. (*$14* засчитывается за 1 символ. Это шестнадцатеричный код для символа Ctrl-T.)



15. Переходите к разделу «Ввод символов ASCII» на стр. 12-21.

16.Введите следующую цепочку:



17.Введите входное условие (условия), которое задаст момент отправки:

- Вы можете использовать любой тип входной инструкции.
- Эта инструкция должна изменять значение с “ложь” на “истина” каждый раз, когда должны быть отправлены данные.

18.Введите 0.

19.Введите имя тега, в котором сохраняются символы ASCII. Определите тип данных как **строковый**.

20.Введите имя тега для инструкции AWA и определите тип данных как SERIAL_PORT_CONTROL.

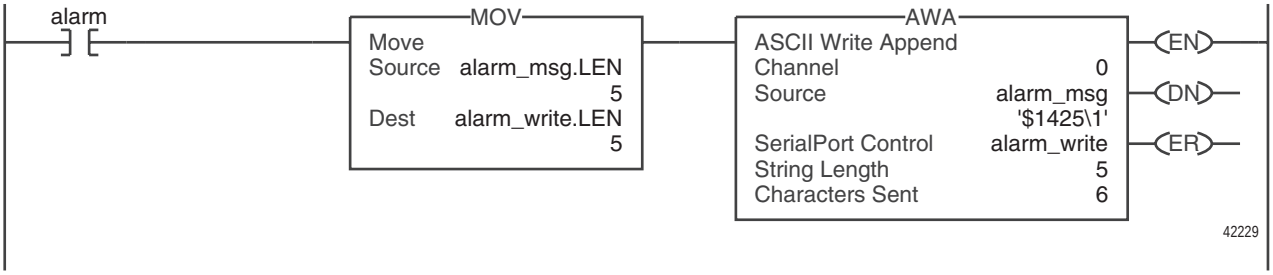
21.Введите член LEN тега Source. (Тег из шага 19.)

22.Введите член LEN инструкции AWA. (Тег из шага 20.)

ПРИМЕР

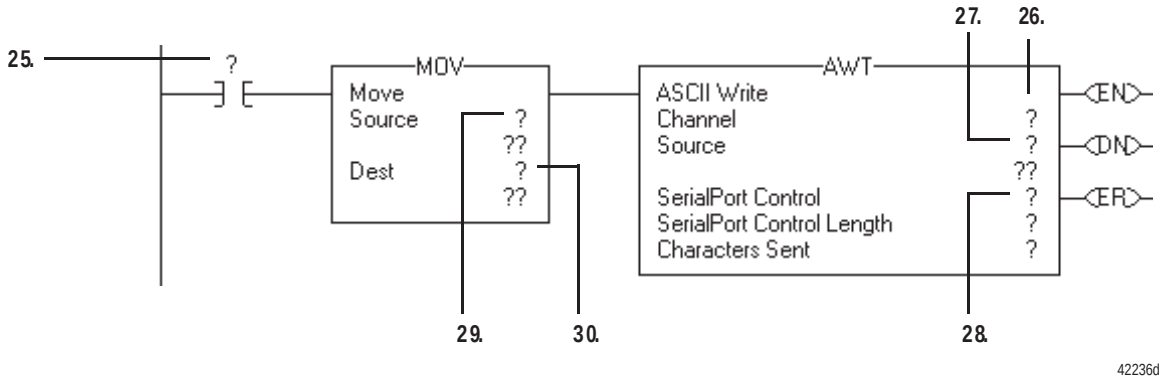
Когда *alarm* установлен, инструкция AWA посылает символы в *alarm_msg* и добавляет символ окончания.

- Поскольку количество символов в *alarm_msg* меняется, цепочка, в первую очередь, перемещает длину *alarm_msg* (*alarm_msg*.LEN) в параметр length инструкции AWA (*alarm_write*.LEN).
- В *alarm_msg* *$14* засчитывается за один символ. Это шестнадцатеричный код для символа Ctrl-T.



23.Переходите к разделу «Ввод символов ASCII» на стр. 12-21.

24.Введите следующую цепочку:



25.Введите входное условие (условия), которое задаст момент отправки:

- Вы можете использовать любой тип входной инструкции.
- Эта инструкция должна изменять значение с “ложь” на “истина” каждый раз, когда должны быть отправлены данные.

26.Введите 0.

27.Введите имя тега, в котором сохраняются символы ASCII. Определите тип данных как **строковый**.

28.Введите имя тега для инструкции AWT и определите тип данных как SERIAL_PORT_CONTROL.

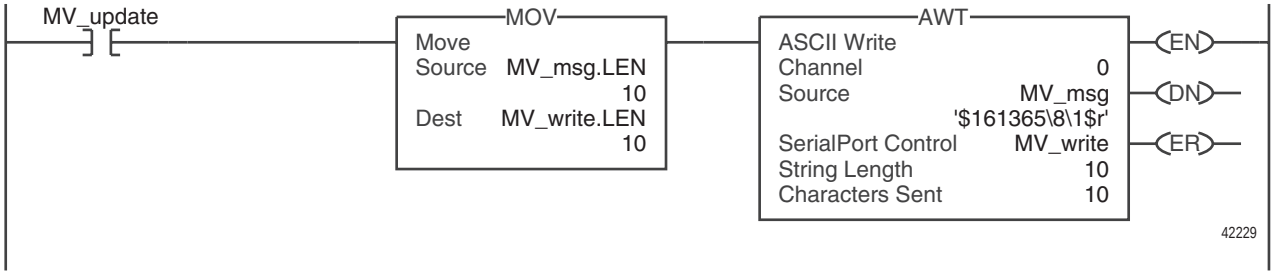
29.Введите член LEN тега Source. (Тег из шага 27.)

30.Введите член LEN инструкции AWT. (Тег из шага 28.)

ПРИМЕР

Когда *MV_update* установлен, инструкция AWT посылает символы в *MV_msg*.

- Поскольку количество символов в *MV_msg* меняется, цепочка, в первую очередь, перемещает длину *MV_msg* (*MV_msg*.LEN) в параметр length инструкции AWT (*MV_write*.LEN).
- В *MV_msg* *$16* засчитывается за один символ. Это шестнадцатеричный код для символа Ctrl-V.



31.Переходите к разделу «Ввод символов ASCII» на стр. 12-21.

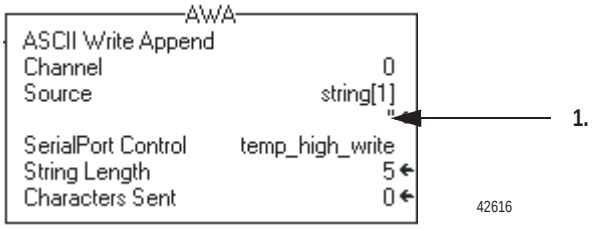
Ввод символов ASCII

Определитесь, нужен ли вам этот шаг:

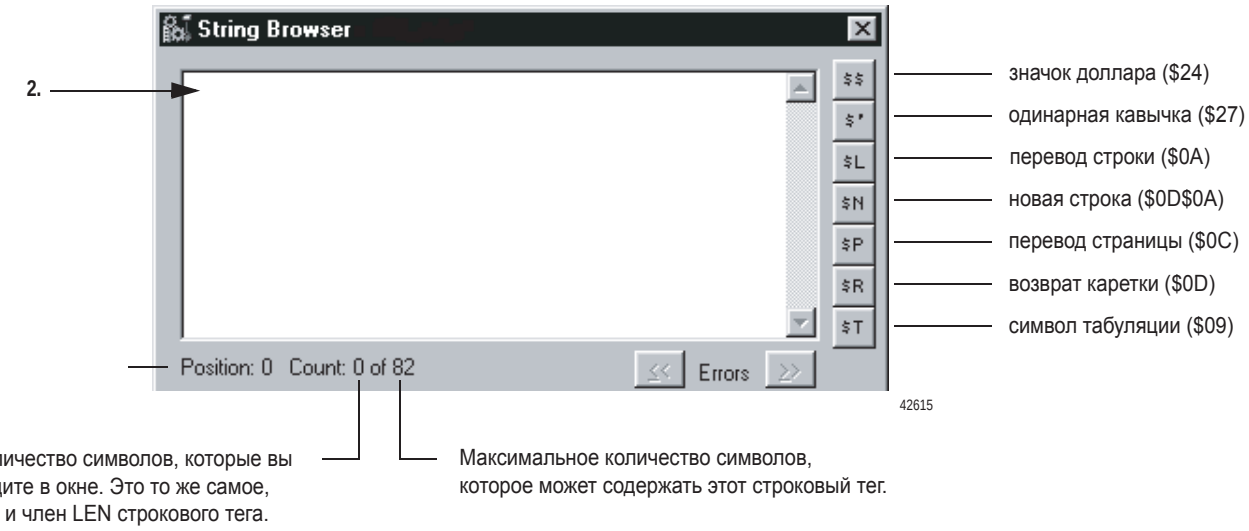
Если:	То:
Вам необходим алгоритм для создания строки.	Переходите к разделу «Обработка символов ASCII» на стр. 13-1.
Вы хотите ввести символы.	Переходите к шагу 1.

ВАЖНО

В окне String Browser (Просмотр строк) показываются символы до значения члена LEN этого строкового тега. Этот строковый тег может содержать дополнительные данные, которые не показываются в окне String Browser.



1. Дважды щелкните в области значений Source.



2. Введите символы строки.

3. Выберите *OK*.

Для заметок:

Обработка символов ASCII

Когда использовать данную процедуру

Используйте данную процедуру для:

- Расшифровки штрихового кода и действий на основе этого штрихового кода
- Использования значения веса, получаемого от весов, в случае, когда для передачи значения веса используются символы ASCII
- Расшифровки сообщений от устройства, запускающегося при помощи символов ASCII, такого как терминал оператора
- Создания строки для устройства, запускающегося при помощи символов ASCII, используя переменные вашего приложения

Как использовать данную процедуру

ВАЖНО Если вы не знакомы с вводом релейной логики в проекте RSLogix 5000, то, в первую очередь, ознакомьтесь с разделом «Программирование на языке релейной логики» на стр. 8-1.

В зависимости от вашего приложения, возможно, вам не понадобиться использовать все задачи данной процедуры. Представленная ниже таблица позволит определить, с чего начать.

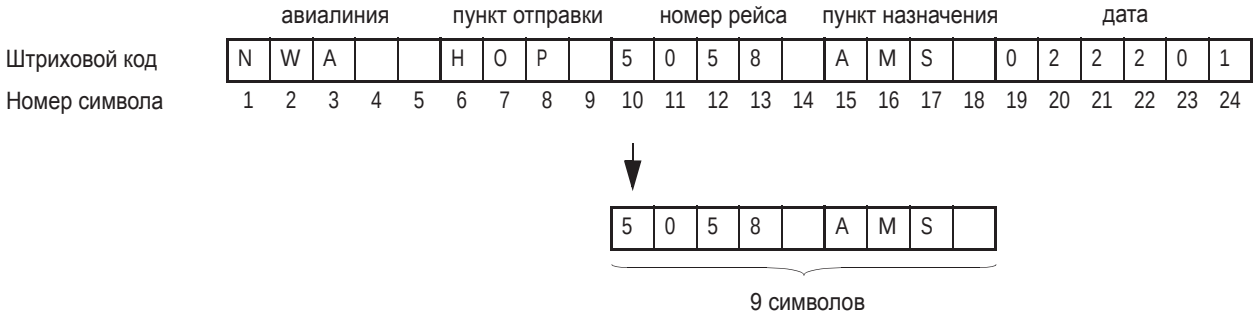
Если вы хотите:	То переходите к разделу:	На стр.
Вычленить часть штрихового кода	Выделение части штрихового кода	13-2
Организовать поиск заданной строки в массиве	Поиск штрихового кода	13-4
Сравнить строки символов	Проверка символов штрихового кода	13-10
Получить вес от весов	Преобразование значений	13-12
Расшифровать сообщение от терминала оператора	Расшифровка сообщения ASCII	13-14
Создать строку для отправки в терминал оператора	Построение строки	13-18

За дополнительной информацией об инструкциях, связанных с символами ASCII, обратитесь к *справочному руководству «Общие инструкции контроллера Logix5000»,* документ 1756-PM003.

Выделение части штрихового кода

Для выделения части штрихового кода, и проведения каких-либо действий на основе этих данных используйте шаги, представленные ниже.

Например, штриховой код может содержать информацию о багаже на конвейере в аэропорту. Для проверки номера рейса и пункта назначения вам необходимо выделить символы 10 - 18.



Шаги

1. Введите следующую цепочку:

2. Введите бит EM инструкции ARD, которая считывает штриховой код.

3. Введите строковый тег, который содержит штриховой код.

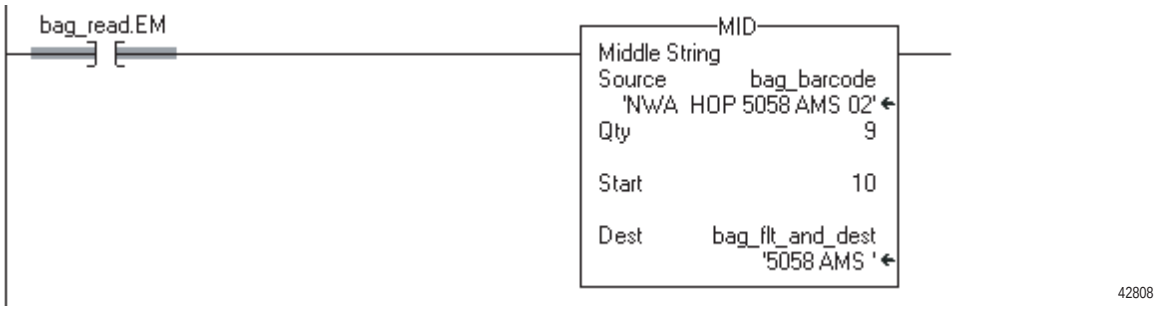
4. Введите количество символов в части штрихового кода, которую вы хотите проверить.

5. Введите позицию первого символа в части штрихового кода, которую вы хотите выделить.

6. Введите имя тега для сохранения части штрихового кода, которую вы хотите проверить. Определите тип данных как строковый.

ПРИМЕР

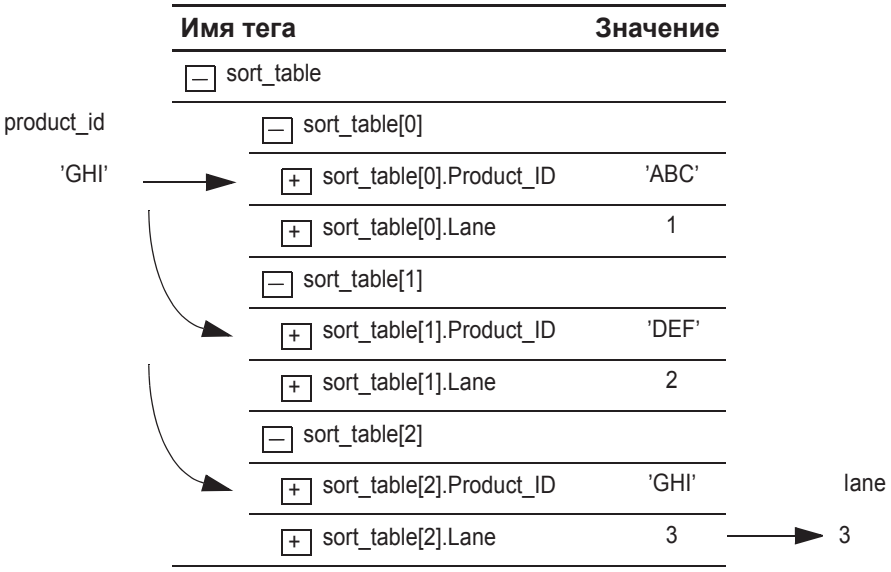
На багажном конвейере аэропорта каждый чемодан имеет штриховой код. Символы 10 – 18 этого штрихового кода являются номером рейса и аэропортом назначения для данного чемодана. После того, как штриховой код считан, (установлен *bag_read*.EM) инструкция MID копирует номер рейса и аэропорт назначения в тег *bag_flt_and_dest.*



Поиск штрихового кода

Используйте представленные ниже шаги для поиска заданной информации на основе штрихового кода.

Например, в операциях по сортировке, массив данных, заданный пользователем, создает таблицу, которая показывает номер маршрута для каждого типа продукта. Чтобы определить, какой маршрут ведет к продукту, контроллер ищет идентификационный номер продукта в данной таблице (символы штрихового кода, которые идентифицируют продукт).

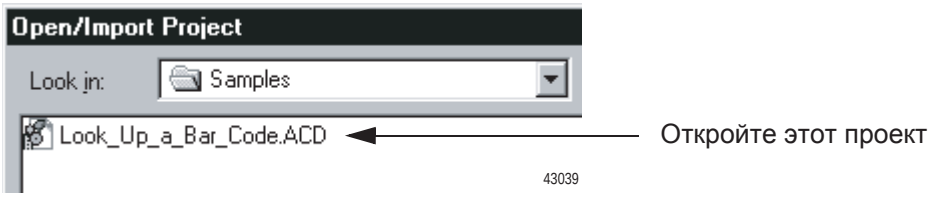


Для поиска штрихового кода:

- Создайте тип данных PRODUCT_INFO
- Осуществите поиск символов
- Идентифицируйте номер маршрута
- Удалите неверные символы
- Введите идентификационный номер продукта и номер маршрута

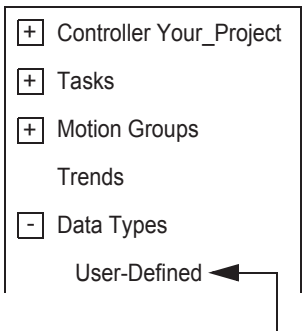
СОВЕТ

Чтобы скопировать элементы из проекта, приведенного в качестве примера, откройте папку *...\RSLogix 5000\Projects\Samples*.



Создание типа данных PRODUCT_INFO

Чтобы создать новый тип данных:

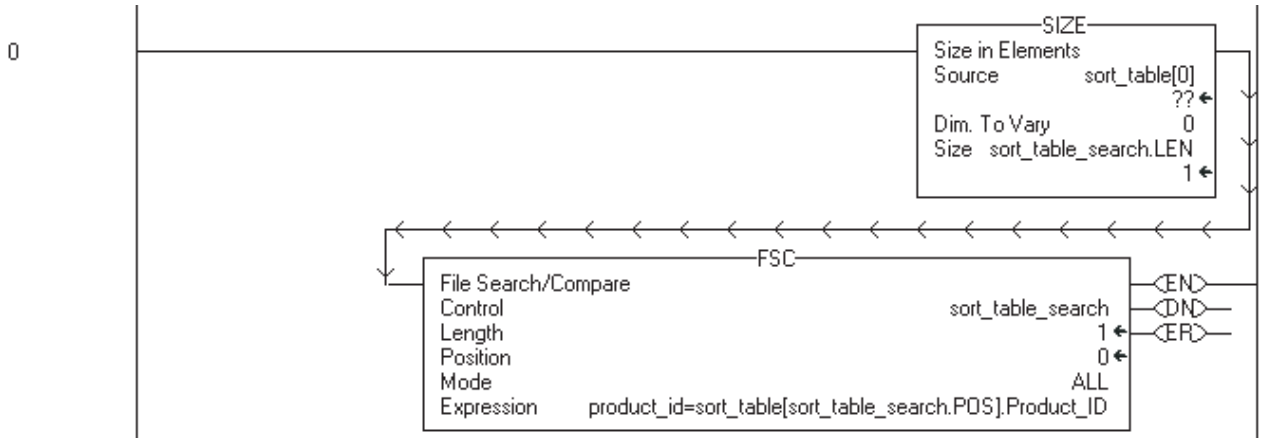


Щелкните правой клавишей мыши и выберите *New Data Type (Новый тип данных)*.

Создайте следующий тип данных, задаваемый пользователем.

Тип данных: PRODUCT_INFO				
Имя	PRODUCT_INFO			
Описание	Определяет пункт назначения для элемента на основе строки символов ASCII, которая идентифицирует этот элемент			
Члены				
	Имя	**Тип данных**	**Формат**	**Описание**
	Product_ID	STRING		Символы ASCII, которые идентифицируют этот элемент
	Lane	DINT	Десятичный	Пункт назначения для элемента на основе его идентификатора

Поиск символов

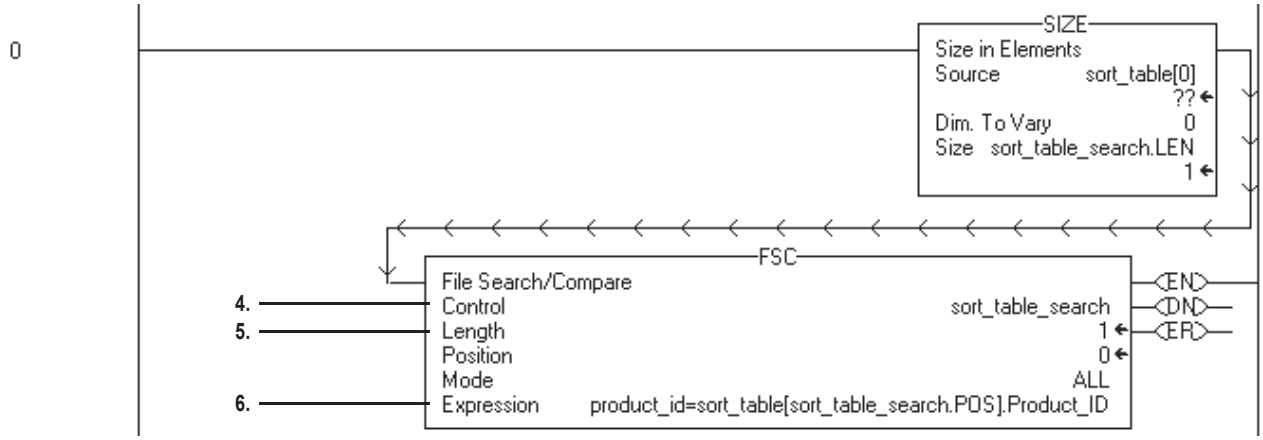


1. Инструкция SIZE подсчитывает количество элементов в массиве *sort_table*. Этот массив содержит идентификатор каждого элемента с соответствующим маршрутом.

Имя тега	Тип
sort_table	PRODUCT_INFO[*number_of_items*] где: *number_of_items* – количество элементов, которые вы хотите сортировать.

2. Инструкция SIZE подсчитывает количество элементов в Dimension 0 массива. В этом случае, это только размерность.
3. Инструкция SIZE задает Length (Длина) последующей инструкции FSC равной размеру массива *sort_table*. Это гарантирует, что инструкция FSC ищет точный размер массива.

Имя тега	Тип
sort_table_search	CONTROL



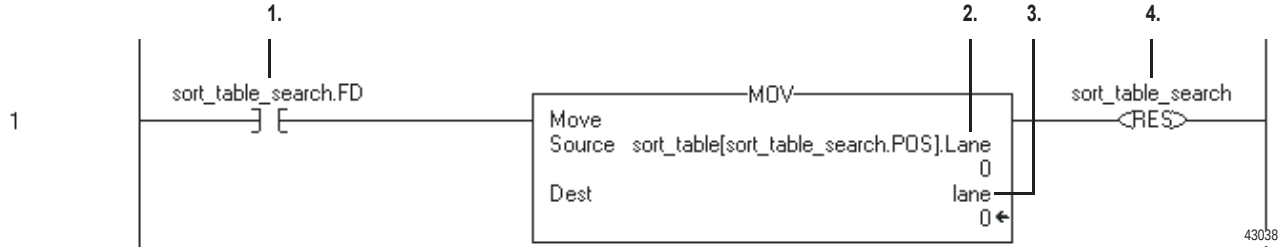
4. Тег *sort_table_search* управляет инструкцией FSC, которая осуществляет поиск в массиве *sort_table* символов штрихового кода.

5. Хотя предыдущая инструкция задает Length (Длину) этой инструкции, программному обеспечению требуется начальное значение для проверки проекта.

6. Тег *product_id* содержит символы штрихового кода, которые идентифицируют данный элемент. Инструкция FSC ищет каждый член Product_ID в массиве *sort_table* ,пока инструкция не находит совпадения с тегом *product_id*.

Имя тега	Тип
product_id	STRING

Идентификация номера маршрута



1. Когда инструкция FSC находит идентификатор продукта в массиве *sort_table*, эта инструкция устанавливает бит FD.
2. Когда FSC находит совпадение, элемент POS указывает номер данного совпадения в массиве *sort_table*. Соответствующий член LANE указывает номер маршрута данного совпадения.
3. На основе значения POS инструкция MOV перемещает соответствующий номер маршрута в тег *lane*. Контроллер использует значение этого тега как маршрут для данного элемента.

Имя тега	Тип
lane	DINT

4. После того, как инструкция MOV задала значение тега *lane*, инструкция RES вернет в исходное положение инструкцию FSC, и она снова может осуществлять поиск идентификатора продукта.

Отбрасывание неверных символов

1. Если инструкция FSC не находит идентификатор продукта в массиве *sort_table*, эта инструкция устанавливает бит DN.
2. Когда соответствие не находится, инструкция MOV перемещает 999 в тег маршрута. Это говорит контроллеру о том, что этот элемент необходимо отбросить или изменить его маршрут.
3. После того, как инструкция MOV задаст значение тега *lane*, Инструкция RES вернет в исходное положение инструкцию FSC, и она снова может осуществлять поиск идентификатора.

Ввод идентификаторов продуктов и номеров маршрутов

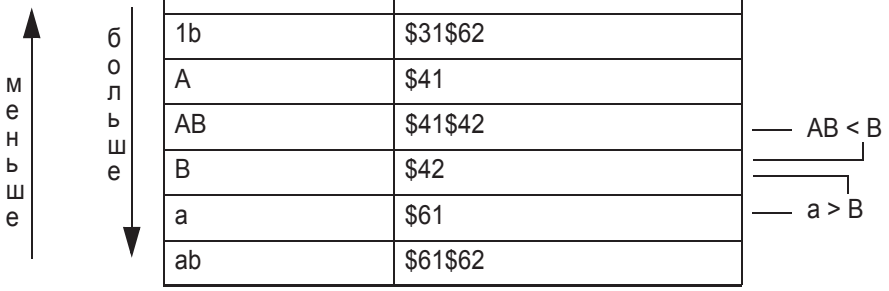
Введите ASCII символы, идентифицирующие каждый элемент, и соответствующие номера маршрутов в массив *sort_table.*

Имя тега	Значение
⊟ sort_table	{...}
⊟ sort_table[0]	{...}
⊞ sort_table[0].Product_ID	символы ASCII, которые идентифицируют первый элемент
⊞ sort_table[0].Lane	номер маршрута для этого элемента
⊟ sort_table[1]	{...}
⊞ sort_table[1].Product_ID	символы ASCII, которые идентифицируют следующий элемент
⊞ sort_table[1].Lane	номер маршрута для этого элемента

Проверка символов штрихового кода

В данной задаче вы используете инструкцию сравнения (EQU, GEQ, GRT, LEQ, LES, NEQ) для проверки заданных символов.

- Шестнадцатеричные значения символов определяют, больше одна строка другой, или меньше.
- Когда две строки сортируются как в телефонном справочнике, порядок следования строк определяется тем, которая из них больше.

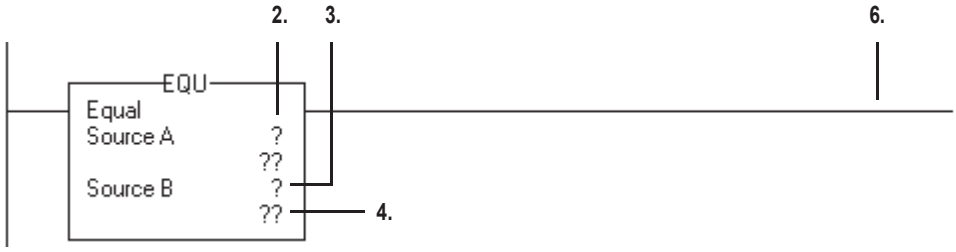


Символы ASCII	Шестнадцатеричные коды
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
ab	$61$62

Шаги:

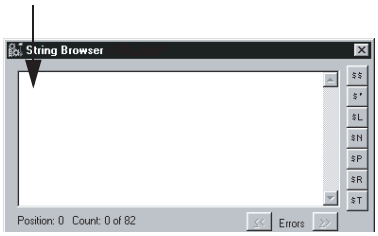
1. Введите цепочку и инструкцию сравнения:

Чтобы узнать, что эта строка:	Введите эту инструкцию:
Равна заданным символам	EQU
Не равна заданным символам	NEQ
Больше чем заданные символы	GRT
Равна или больше чем заданные символы	GEQ
Меньше чем заданные символы	LES
Равна или меньше чем заданные символы	LEQ



2. Введите тег, который хранит часть штрихового кода, которую вы хотите проверить. (Destination из Extract a Part of a Bar Code, шаг 6.)

3. Введите имя тега, который хранит символы, относительно которых вы хотите проводить проверку. Определите тип данных как строковый.



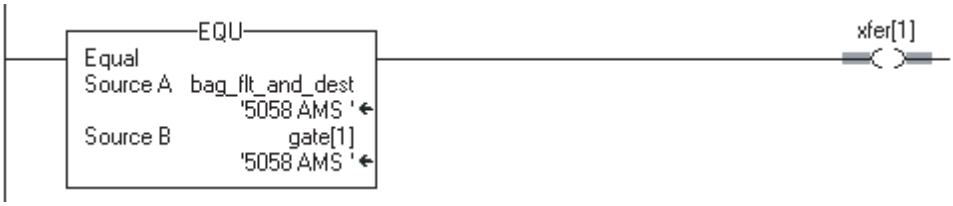
4. Дважды щелкните в области значения источника B (Source B).

5. Введите символы ASCII, относительно который вы проводите проверку, и выберите OK.

6. Введите требуемый выход.

ПРИМЕР

Если *bag_flt_and_dest* равняется *gate[1]*, то *xfer[1]* включается, что направляет чемодан в нужную дверь.



7. Вы хотите проверить другую часть штрихового кода?

Если:	То:
Да	Переходите к разделу «Выделение части штрихового кода» на стр. 13-2
Нет	Стоп. Вы выполнили данную процедуру.

Преобразование значений

Используйте указанные ниже шаги для преобразования представлений значений ASCII в значения DINT или REAL для использования в вашем приложении.

- Инструкции STOD и STOR пропускают любые начальные символы управления или нечисловые символы (за исключением знака минус перед числом).
- Если строка содержит насколько групп чисел, которые разделены разделителями данных (напр. /), инструкции STOD и STOR преобразуют только первую группу.

Шаги:

1. Какой тип имеет значение?

Если:	То:
С плавающей точкой	Переходите к шагу 2.
Целый	Переходите к шагу 7.

2. Введите следующую цепочку:

3. Введите бит EM инструкции ARD или ARL, который считывает значение.

4. Введите строковый тег, который содержит значение.

5. Введите имя тега для сохранения этого значения для использования в вашем приложении. Определите тип данных как REAL.

ПРИМЕР

После считывания значения веса из весов (устанавливается *weight_read*.EM), инструкция STOR преобразует цифровые символы из *weight_ascii* в значение типа REAL и сохраняет результат в *weight*.

6. Переходите к шагу 11.

7. Введите следующую цепочку:

8. Введите бит EM инструкции ARD или ARL, который считывает значение.

9. Введите строковый тег, который содержит значение.

10. Введите имя тега для сохранения этого значения для использования в вашем приложении. Определите тип данных как DINT.

ПРИМЕР

Когда установлен *MV_read.EM*, инструкция STOD преобразует первый набор цифровых символов из тега *MV_msg* в целое значение. Инструкция пропускает начальный символ управления ($06) и останавливается на разделителе (\ ).

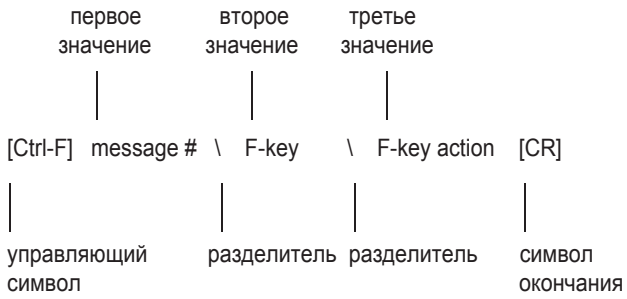
11. Содержит ли данная строка еще одно значение, которое вы хотите использовать?

Если:	То:
Да	Переходите к разделу «Расшифровка сообщения ASCII» на стр. 13-14.
Нет	Стоп. Вы выполнили данную процедуру.

Расшифровка сообщения ASCII

Для выделения и преобразования значения из сообщения ASCII, содержащего несколько значений, используйте следующие шаги.

Например, сообщение может выглядеть следующим образом:



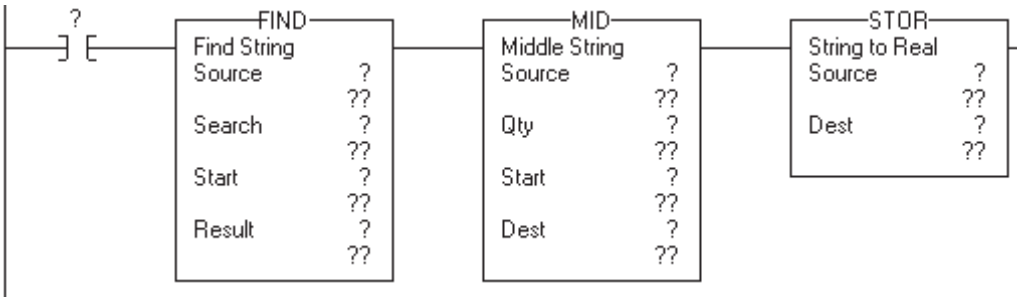
1. Определите, с чего начать:

Если:	И:	То:
Строка содержит более одного значения	Это первое значение.	Переходите к разделу «Преобразование значений» на стр. 13-12.
	Это не первое значение.	Переходите к шагу 2.
Строка содержит одно значение	→	Переходите к разделу «Преобразование значений» на стр. 13-12.

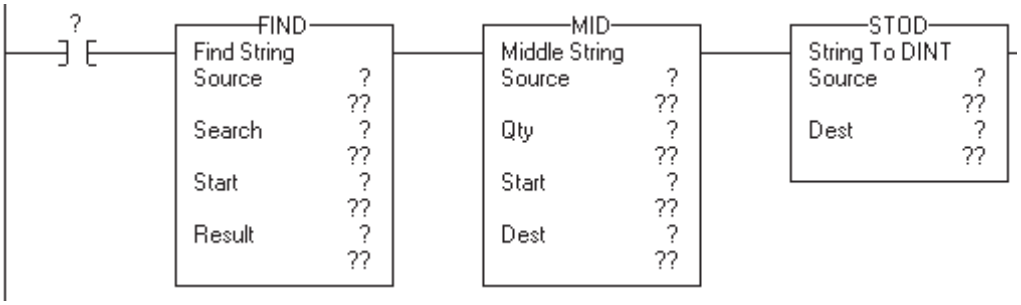
2. Какого типа данное значение?

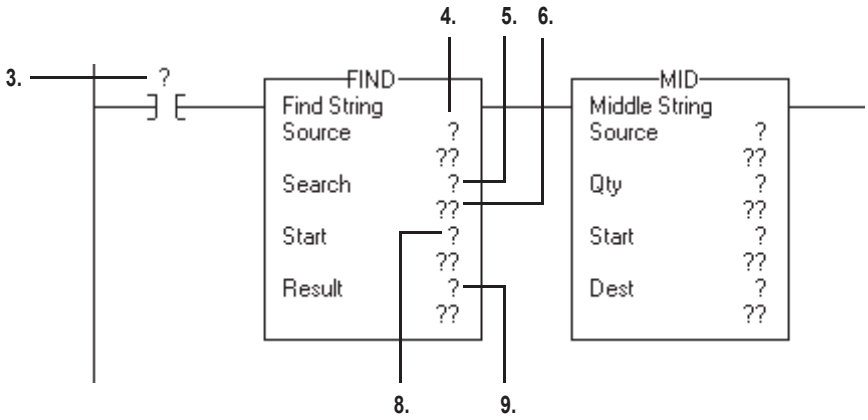
Если:	То:
С плавающей точкой	Введите цепочку A: Поиск и преобразование значения с плавающей точкой (Find and Convert a Floating-Point Value)
Целое	Введите цепочку B: Поиск и преобразование целого значения (Find and Convert an Integer Value)

Цепочка A: Поиск и преобразование значения с плавающей точкой



Цепочка B: Поиск и преобразование целого значения



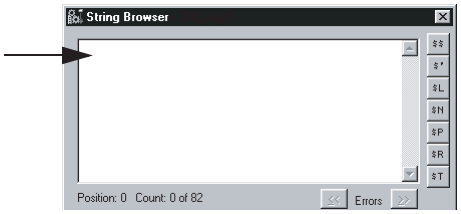


3. Введите бит EM инструкции ARL, который считывает значение.

4. Введите строковый тег, который содержит значение.

5. Введите имя тега для сохранения разделителя, который помечает начало данного значения. Определите тип данных как строковый.

6. Дважды щелкните в области значений Search (Поиск).

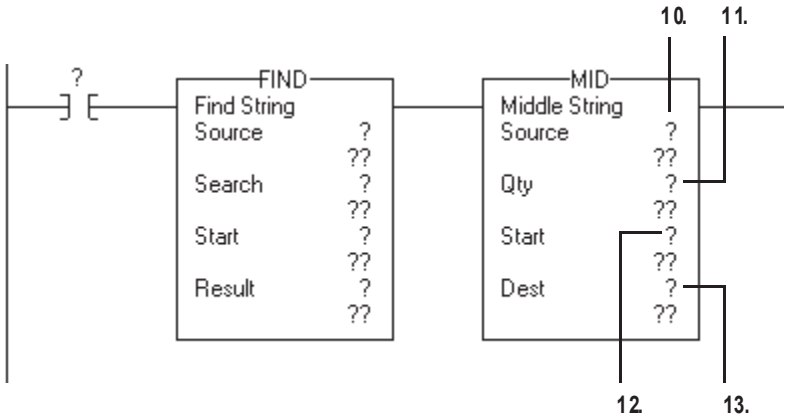


7. Введите разделитель и выберите *OK*.

8. Введите позицию в строке для начала поиска.

- Первоначально, вы можете использовать 0 для поиска первого разделителя.
- Для расшифровки дополнительных данных, увеличьте это значение для поиска следующего разделителя.

9. Введите имя тега для сохранения положения разделителя. Определите тип данных как DINT.



10.Введите строковый тег, который содержит данное значение.

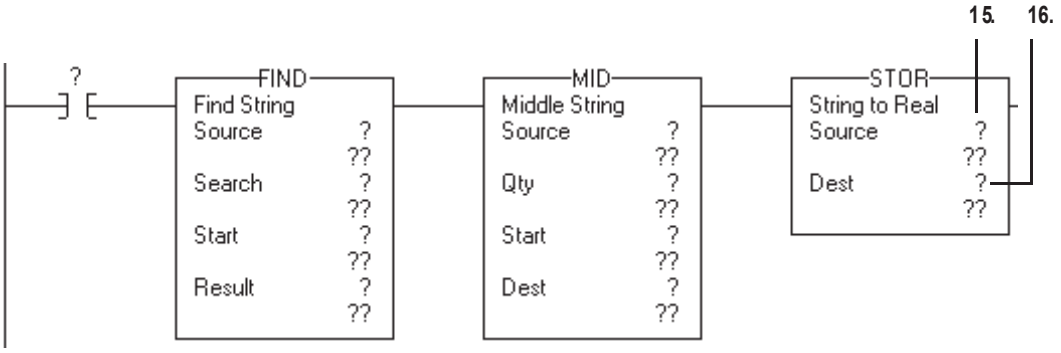
11.Введите максимальное количество символов, которое может содержать данное значение.

12.Введите тег, который сохраняет положение делителя. (Тег 9-го шага.)

13.Введите имя тега для сохранения данного значения. Определите тип данных как строковый.

14.Какую инструкцию преобразования вы использовали?

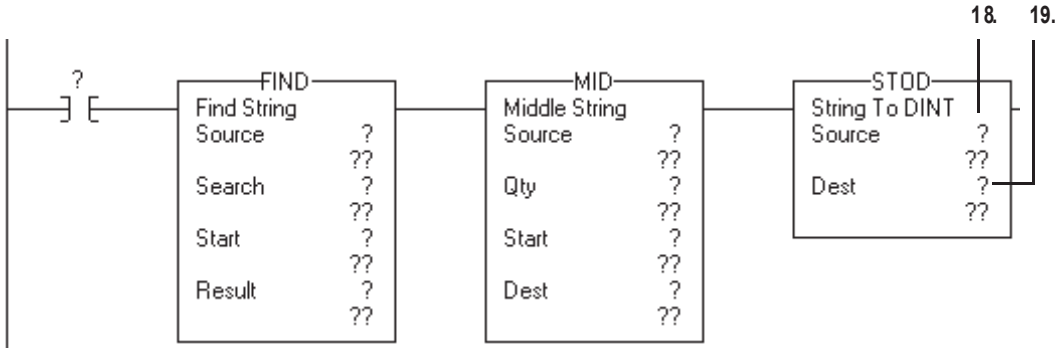
Если:	**То:**
STOR	Переходите к шагу 15.
STOD	Переходите к шагу 18.



15.Введите тег, который сохраняет данное значение. (Тег шага 13.)

16.Введите имя тега для сохранения этого значения для использования в вашем приложении. Определите тип данных как REAL.

17.Переходите к шагу 20.



18.Введите тег, который сохраняет данное значение. (Тег шага 13.)

19.Введите имя тега для сохранения этого значения для использования в вашем приложении. Определите тип данных как DINT.

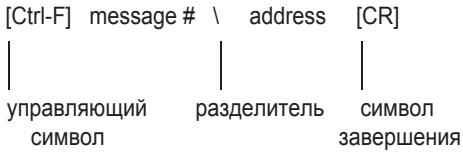
20.Содержит ли данная строка еще одно значение, которое вы хотите использовать?

Если:	**То:**
Да	A. Добавьте 1 к Result инструкции Find. (Тег из шага 9.) B. Повторите шаги 2 - 19.
Нет	Стоп. Вы выполнили данную процедуру.

Построение строки

Для построения строки из переменных вашего приложения используйте представленные ниже шаги. После этого, вы можете послать эту строку в устройство, включаемое при помощи символов ASCII, такое как терминал MessageView.

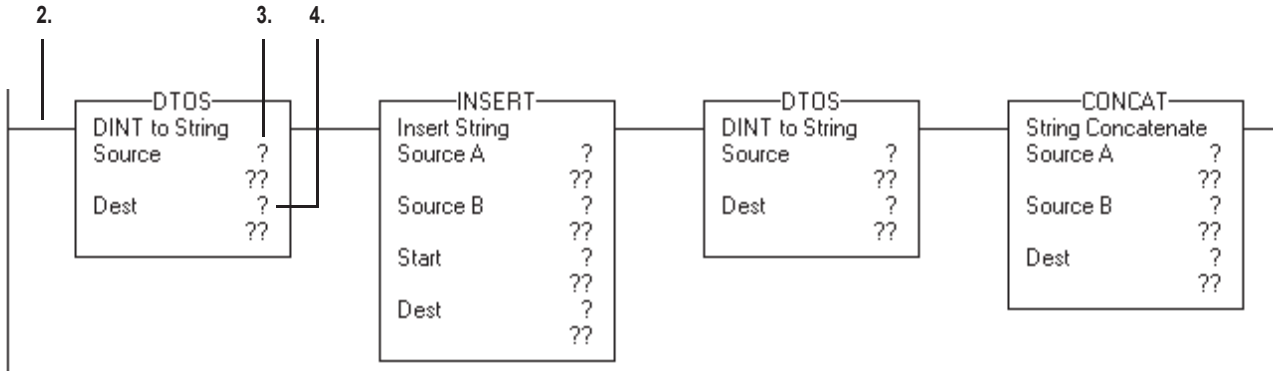
- В данной процедуре вы строите строку, содержащую две переменные. Например, для терминала оператора может потребоваться строка, которая выглядит следующим образом:



- Если вам необходимо включить больше переменных, используйте дополнительные инструкции INSERT или CONCAT.
- Если вам необходимо послать значение с плавающей точкой, то вместо инструкции DTOS используйте инструкцию RTOS.
- Итоговая строка не будет включать символ завершения. При отправке строк используйте инструкцию AWA для автоматического добавления символа окончания.

Шаги:

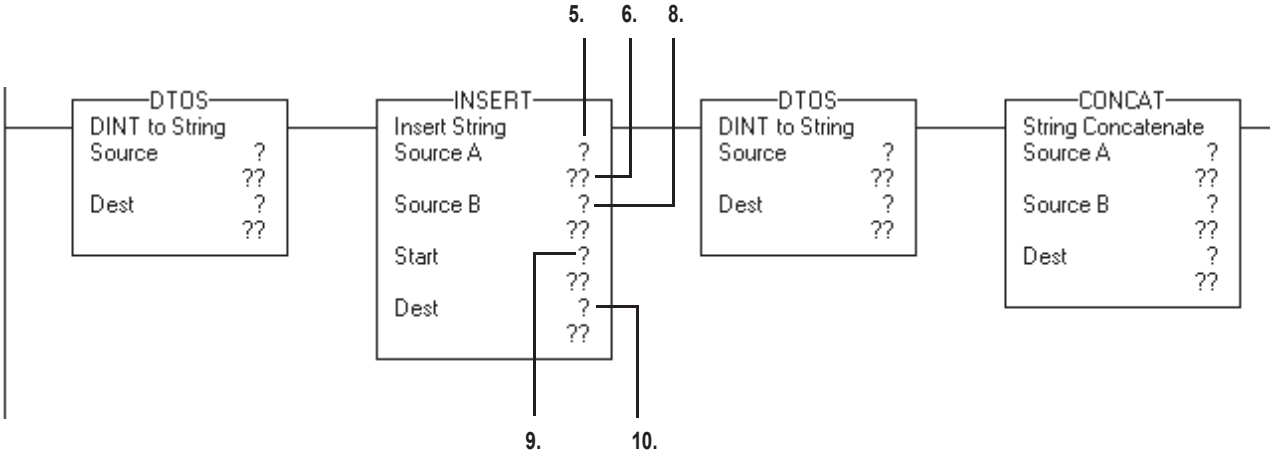
1. Введите следующую цепочку:



2. Введите входное условие (условия) которое задаст, когда необходимо создать строку.

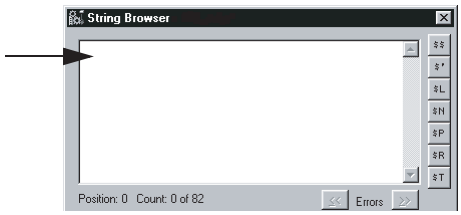
3. Введите тег типа DINT, который содержит первое значение для строки.

4. Введите имя тега для сохранения представления ASCII данного значения. Определите тип данных как строковый.



5. Введите имя тега для сохранения управляющего символа и разделителя для этой строки. Определите тип данных как строковый.

6. Дважды щелкните в области значения Source A.



7. Введите с клавиатуры управляющий символ и разделитель, выберите *OK*.

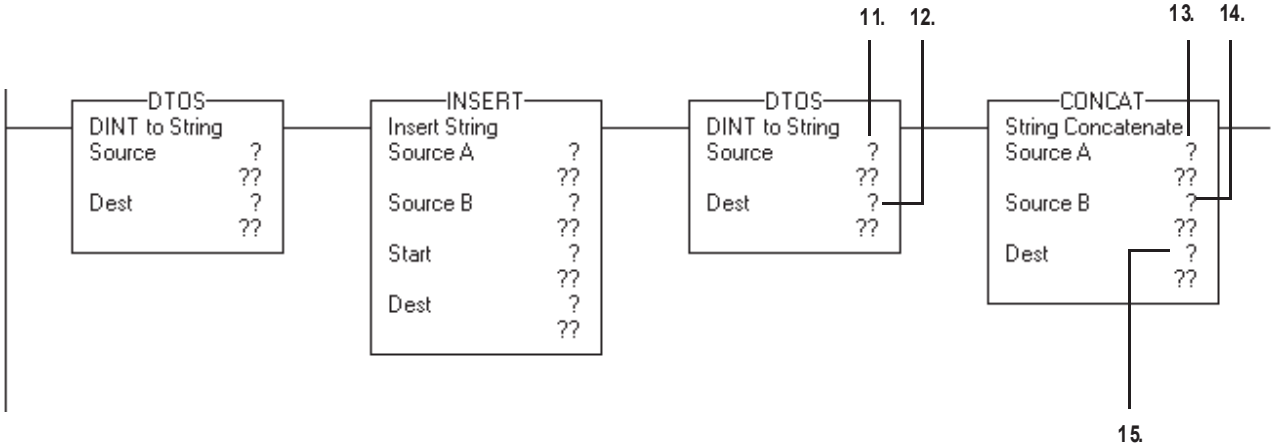
Для управляющего символа введите шестнадцатеричный код. Список шестнадцатеричный кодов представлен на задней обложке данного руководства.

8. Введите тег, который сохраняет представление ASCII первого значения. (Тег шага 4.)

9. Введите 2.

Это поставит данное значение после первого символа (управляющего символа) в Source A.

10. Введите имя тега для сохранения частично созданной строки. Определите тип данных как строковый.



11.Введите тег типа DINT, содержащий второе значение для строки.

12.Введите тег, который сохраняет представление ASCII этого значения. Определите тип данных как строковый.

13.Введите тег, который сохраняет частично созданную строку. (Тег шага 10.)

14.Введите тег, который сохраняет представление ASCII второго значения. (Тег шага 12.)

15.Введите имя тега для сохранения готовой строки. Определите тип данных как строковый.

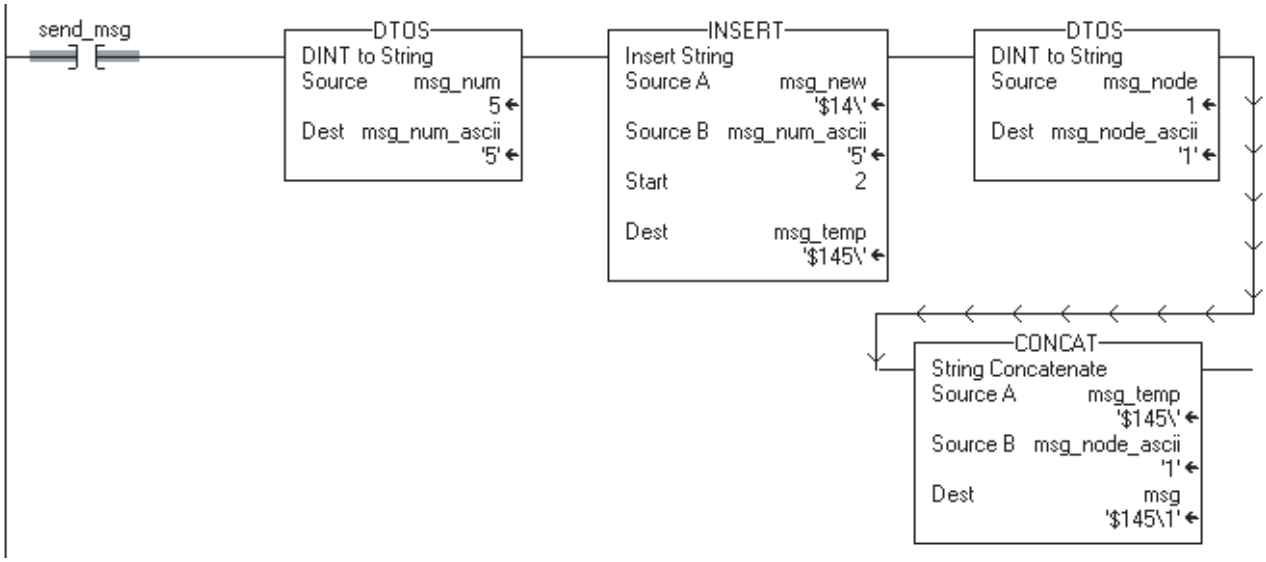
ПРИМЕР

Для запуска сообщения в терминале MessageView, контроллер посылает в терминал сообщение в следующем формате: [Ctrl-T] message # \ address [CR] ([Ctrl-T] номер сообщения \ адрес [CR]).

Когда *send_msg* установлен, цепочка делает следующее:

- Первая инструкция DTOS преобразует номер сообщения в символы ASCII.
- Инструкция INSERT вставляет номер сообщения (в ASCII) за управляющим символом [Ctrl-T]. (Шестнадцатеричный код для Ctrl-T это $14.)
- Вторая инструкция DTOS преобразует номер узла терминала в символы ASCII.
- Инструкция CONCAT ставит номер узла (в ASCII) за обратной косой чертой [\ ] и сохраняет готовую строку в *msg*.

Для отправки сообщения, инструкция AWA отправляет тег *msg* и добавляет символ возврата каретки [CR].



Для заметок:

Форсировка элементов релейной логики

Когда использовать данную процедуру

Используйте форсирову для замены данных, которые используются или производятся вашим алгоритмом. Используйте форсировку, например, в следующих ситуациях:

- При проверке и отладке вашего алгоритма,
- При проверке связи с выходным устройством,
- Для временного сохранения функционирования процесса, когда входное устройство уже дало сбой.

Используйте форсировку только как временную меру. Форсировка не предназначена для постоянной работы в составе вашего приложения.

Как использовать данную процедуру

Если вы хотите:	См.
Ознакомится с предосторожностями, которые вам следует принять при добавлении, изменении, удалении или разрешении форсировок	«Меры предосторожности» на стр. 14-2
Определить текущее состояние форсировок в вашем проекте	«Проверка состояния форсировок» на стр. 14-4
Определить, для какого типа элементов работает форсировка в вашем проекте	«Что форсируется» на стр. 14-6
Ознакомиться с общей информацией о форсировках ввода/вывода, для каких элементов они разрешены и как форсировка ввода/вывода влияет на ваш проект	«Когда использовать форсировку ввода/вывода» на стр. 14-6
Форсировать значения ввода/вывода	«Добавление форсировки ввода/вывода» на стр. 14-8
Ознакомиться с общей информацией о проходе через переход или параллельный путь	«Когда использовать проход» на стр. 14-9
Выполнить проход через активный переход	«Проход через переход или форсировку пути» на стр. 14-9
Выполнить проход через параллельный путь, имеющий значение форсировки «ложь»	
Ознакомиться с общей информацией о форсировках ПФС, для каких элементов они разрешены и как форсировки влияют на ваши ПФС	«Когда использовать форсировку ПФС» на стр. 14-9
Форсировать переход или параллельный путь в ПФС	«Добавление форсировки ПФС » на стр. 14-12
Остановить действие форсировки	«Удаление или запрещение форсировок» на стр. 14-13

Меры предосторожности

Принимайте следующие меры предосторожности при использовании форсировок:

ВНИМАНИЕ

Форсировка может привести к неожиданному движению механизмов, что может повлечь за собой травмы персонала. Перед использованием форсировки определите, как она повлияет на ваш механизм или процесс, и удалите людей из зоны действия механизмов.

- Разрешение форсировки ввода/вывода вызывает изменение входных, выходных, произведенных или потребленных значений.
- Разрешение форсировки ПФС вызывает переход механизма или процесса в другое состояние или фазу.
- Удаление форсировок может, тем не менее, сохранить состояние разрешения форсировок.
- Если форсировки разрешены, и вы устанавливаете форсировку, она действует немедленно.

Разрешение форсировок

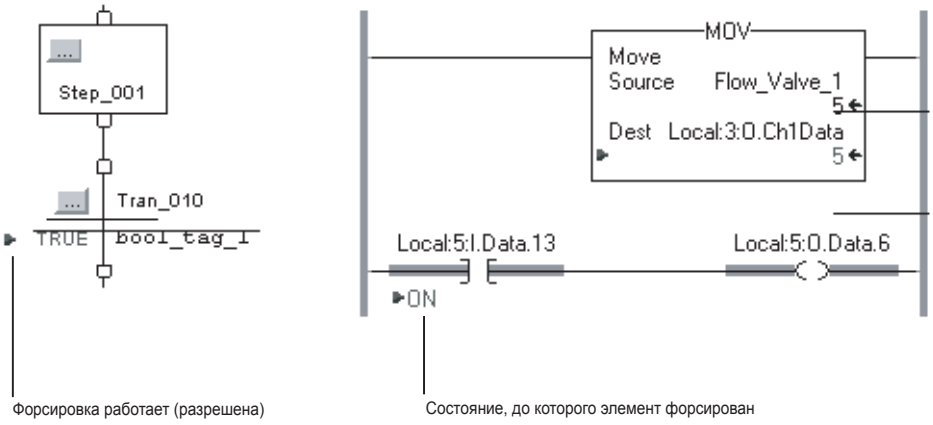
Для того чтобы форсировка заработала, она должна быть разрешена. Вы можете разрешить или запретить форсировку только на уровне контроллера.

- Вы можете разрешить форсировки ввода/вывода и ПФС отдельно или одновременно.
- Вы не можете разрешать или запрещать форсировки для отдельного модуля, набора тегов или элемента тега.

ВАЖНО

Если вы загружаете проект, в котором есть разрешенные форсировки, программное обеспечение предложит вам разрешить или запретить режим форсировки после окончания загрузки.

Когда форсировка работает (разрешена), значок ▶ появляется рядом с форсированным элементом.



Запрещение или удаление форсировки

Для выключения форсировки и выполнения проекта в запрограммированном режиме, запретите или удалите форсировку.

- Вы можете запретить или удалить форсировки ввода/вывода и ПФС отдельно или одновременно.
- Удаление форсировки для тега псевдонима удаляет форсировку для базового тега.

ВНИМАНИЕ

Изменение форсировок может привести к неожиданному движению механизмов, что может повлечь за собой травмы персонала. Перед использованием форсировки определите, как она повлияет на ваш механизм или процесс, и удалите людей из зоны действия механизмов.

Проверка состояния форсировок

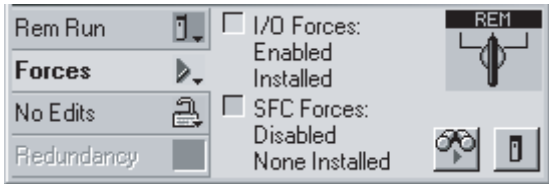
Перед использованием форсировки проверьте состояние форсировок контроллера. Вы можете проверить состояние форсировок следующими способами:

Для определения состояния:	Используйте:
форсировки ввода\вывода	• Панель инструментов Online • Светодиод FORCE • Инструкцию GSV
форсировки ПФС	Панель инструментов Online

Панель инструментов Online

Панель инструментов Online демонстрирует состояние фосировок. Форсировки ввода\вывода и форсировки ПФС показываются отдельно.

Закладка Forces (Форсировки) →



Это:	Означает:
Enabled (Разрешены)	• Если проект содержит форсировки, они *подменяют* алгоритм. • Если вы добавляете форсировку, новая форсировка начинает действовать немедленно.
Disabled (Запрешены)	Форсировки не активны. Если проект содержит форсировки, они *не подменяют* алгоритм.
Installed (Установлены)	По крайней мере, одна форсировка данного типа содержится в данном проекте.
None Installed (Не установлены)	В проекте нет форсировок данного типа.

Светодиод FORCE

Если ваш контроллер оборудован светодиодом FORCE , используйте его для определения состояния любых форсировок ввода\вывода.

ВАЖНО Светодиод FORCE демонстрирует состояние только форсировок ввода\вывода. Форсировки ПФС не показываются.

Если светодиод FORCE:	То:
не горит	• Теги не содержат значения форсировок. • Форсировки ввода\вывода не активны (запрещены).
мигает	• По крайней мере один тег содержит значение форсировки. • Форсировки ввода\вывода не активны (запрещены).
горит не мигая	• Форсировки ввода\вывода активны (разрешены). • Значения форсировок могут существовать, а могут и не существовать.

Инструкция GSV

ВАЖНО Атрибут ForceStatus показывает состояние только форсировок ввода\вывода. Форсировки ПФС не показываются.

Следующий ниже пример показывает, как использовать инструкцию GSV для получения состояния форсировок.

где:

Force_Status – это тег типа DINT.

Для определения что:	Проверьте этот бит:	На наличие значения:
Форсировки установлены (installed)	0	1
Нет установленных форсировок	0	0
Форсировки разрешены	1	1
Форсировки запрещены	1	0

Что форсировать

Вы можете форсировать следующие элементы проекта:

Если вы хотите:	То:
Заменить входное значение, выходное значение, произведенный тег или потребленный тег.	Добавьте форсировку ввода\вывода
Разово заменить условия перехода, чтобы перейти от активного шага к следующему шагу.	Пройдите через переход или форсировки пути.
Разово отменить форсировку параллельного пути и выполнить шаги этого пути.	
Заменить условия перехода в ПФС	Добавьте форсировку ПФС
Выполнить что-либо, но не все пути одновременной ветви ПФС	

Когда использовать форсировку ввода\вывода

Используйте фосировку ввода\вывода для следующих целей:

- Подмены входного значения от другого контроллера (т.е. потребленного тега).
- Подмены входного значения от устройства ввода.
- Подмены логики и задания выходного значения, предназначенного для другого контроллера (т.е. произведенного тега).
- Подмены вашей логики и задания состояния выходного устройства.

ВАЖНО Форсировка увеличивает время выполнения логики. Чем большее количество значений используют форсировку, тем дольше выполняется алгоритм.

ВАЖНО Форсировки хранятся в контроллере, а не в рабочей станции, с которой ведется программирование. Форсировка сохраняется, даже если рабочая станция будет отключена.

Когда используется форсировка ввода\вывода:

- Вы можете вводить форсировку для данных ввода\вывода, за исключением данных по конфигурации.
- Если тег является массивом или конструкцией, такой как тег ввода\вывода, форсируйте элементы или члены типа BOOL, SINT, INT, DINT или REAL.
- Если данные имеют тип SINT, INT или DINT, вы можете форсировать все значение или отдельные биты в пределах значения. Отдельные биты могут иметь следующие состояния форсировки:
  - no force (нет форсировки)
  - force on (форсировка включена)
  - force off (форсировка отключена).

- Вы можете вводить форсировку для псевдонима элемента структуры ввода\вывода, произведенного тега или потребленного тега.
  - Тег-псевдоним использует значения данных совместно с базовым тегом, поэтому форсировка тега-псевдонима вызывает форсировку связанного с ним базового тега.
  - Снятие форсировки с тега-псевдонима снимает форсировку с соответствующего базового тега.

Форсировка входного значения

Форсировка входного или потребленного тега:

- Приводит к замене значения независимо от физического устройства или произведенного тега.
- Не влияет на значение, получаемое другими контроллерами, отслеживающими этот вход или произведенный тег.

Форсировка выходного значения

Форсировка выхода или произведенного тега заменяет логику для физического устройства или другого контроллера (контроллеров). Другие контроллеры, отслеживающие этот модуль выхода в режиме слежения, так же увидят это форсированное значение.

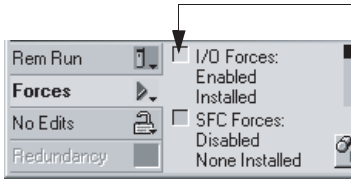
Добавление форсировки ввода\вывода

Используйте форсировку для замены входного значения, выходного значения, произведенного тега и потребленного тега.

ВНИМАНИЕ

Форсировка может привести к неожиданному движению механизмов, что может повлечь за собой травмы персонала. Перед использованием форсировки определите, как она повлияет на ваш механизм или процесс, и удалите людей из зоны действия механизмов.

- Разрешение форсировки ввода/вывода вызывает изменение входных, выходных, произведенных и потребленных значений.
- Если форсировка разрешена, она начинает действовать немедленно.



1. Каково состояние индикатора форсировки ввода вывода?

Если:	Тогда:
не горит	В данный момент не существует форсировок ввода\вывода.
мигает	Форсировки ввода\вывода не активны. Но по крайней мере одна форсировка существует в вашем проекте. Когда вы разрешиете форсировки ввода/вывода, Начнут действовать все форсировки ввода/вывода.
горит не мигая	Форсировки ввода\вывода разрешены (активны). Когда вы устанавливаете (добавляете) форсировку, она начинает действовать немедленно.

2. Откройте процедуру, содержащую тег, который вы хотите форсировать.

3. Щелкните правой клавишей мыши на этом теге и выберите *Monitor...* Если необходимо, раскройте этот тег, чтобы увидеть значение, которое вы хотите форсировать (напр. Значение BOOL тега типа DINT).

4. Установите значение форсировки:

Для форсировки:	Сделайте следующее:
Значения типа BOOL	Щелкните правой клавишей мыши на теге и выберите *Force ON* или *Force OFF*
Значения типа не BOOL	В колонке *Force Mask* для данного тега введите с клавиатуры значение, на которое вы хотите форсировать данный тег. Затем нажмите клавишу *Enter*.

5. Разрешены ли форсировки ввода/вывода? (См. Шаг 1.)

Если:	То:
нет	В меню *Logic(Логика)* выберите *I/O Forcing >Enable All I/O Forces*. Затем выберите *Yes* для подтверждения.
да	Стоп.

Когда использовать проход

Чтобы использовать опцию *Step Through (Проход)* для разовой замены значения «ложь» перехода и выхода из активного шага в следующий шаг:

- У вас нет необходимости добавлять, разрешать, запрещать или удалять форсировки.
- В следующий раз, когда ПФС достигнет данного перехода, он выполнится в соответствиями с условиями перехода.

Эта опция позволяет вам проводить разовую замену значения «ложь» параллельного пути. Когда вы обходите форсировку, ПФС выполняет шаги этого пути.

Проход через переход или форсировку пути

Чтобы обойти переход активного шага или форсировку параллельного пути:

1. Откройте процедуру ПФС.
2. Щелкните правой клавишей мыши на переходе или пути, который форсирован и выберите *Step Through*.

Когда использовать форсировку ПФС

Для замены логики ПФС у вас имеются следующие опции:

Если вы хотите:	То:
Изменять условия перехода всякий раз, как ПФС достигает этого перехода.	Форсируйте переход
Предотвращать выполнение одного или более путей одновременной ветви.	Форсируйте параллельный путь

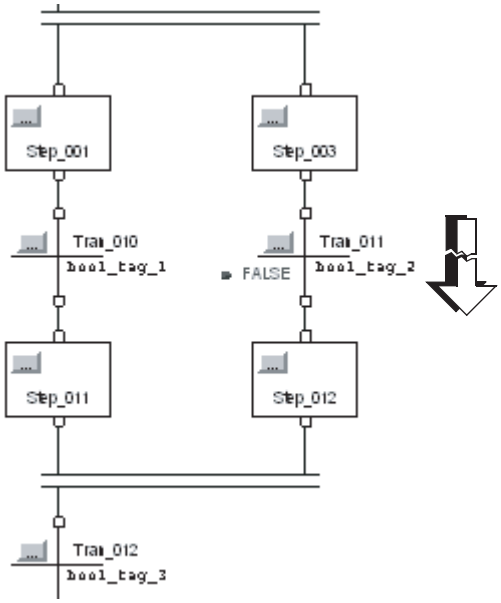
Форсировка перехода

Для подмены условий перехода через повторные выполнения ПФС, форсируйте переход. Форсировка сохраняется, пока вы ее не удалите или не запретите форсировки.

Если вы хотите:	То:
Предотвратить переход ПФС на следующий шаг.	Форсируйте значение перехода на «ложь».
Заставить ПФС перейти на следующий шаг независимо от условий перехода.	Форсируйте значение перехода на «истина».

Если вы форсируете переход внутри одновременной ветви на значение «ложь», ПФС остается в этой одновременной ветви так долго, пока форсировка остается активной (установленной или разрешенной).

- Для того, чтобы выйти из одновременной ветви , последний шаг пути должен быть выполнен, по крайней мере, один раз и переход ниже этой ветви должен иметь значение «истина».
- Форсировка перехода на значение «ложь» предотвращает возможность того, что ПФС достигнет последнего шага пути.
- Когда вы удалите или запретите форсировку, ПФС сможет выполнить оставшиеся шаги этого пути.

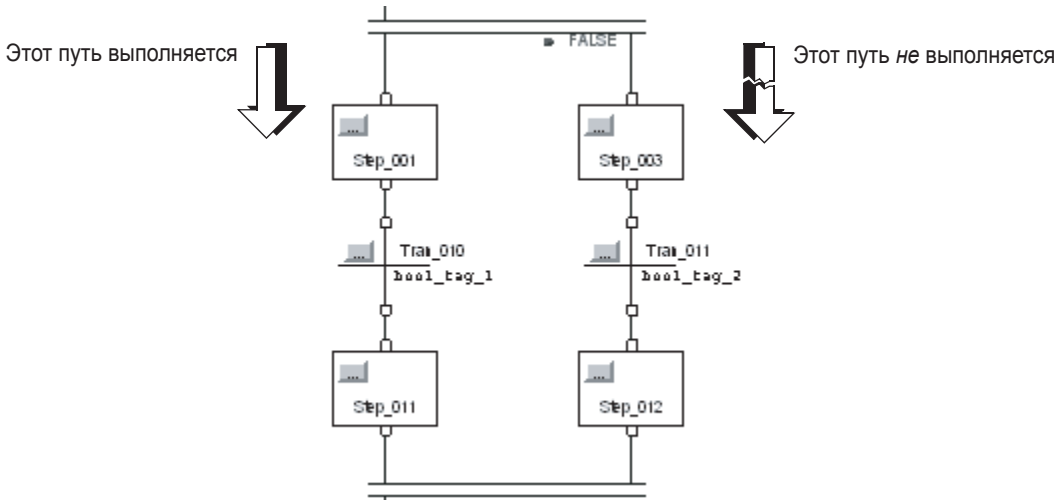


Например, для выхода из этой ветви ПФС должна быть способна:

- Выполнить *Step_011* хотя бы один раз
- Получить последнее значение *Tran_011* и выполнить *Step_012* хотя бы один раз
- Определить, что имеется *Tran_012*

Форсировка параллельного пути

Для предотвращения выполнения пути одновременной ветви, форсируйте значение на «ложь». Когда ПФС достигнет этой ветви, она выполнить только нефорсированные пути.



Если вы форсируете путь одновременной ветви на значение «ложь», ПФС останется в этой ветви пока форсировка остается активной (установленной или разрешенной).

- Для того, чтобы уйти из одновременной ветви, последний шаг каждого пути должен быть выполнен хотя бы один раз и переход ниже этой ветви должен иметь значение «истина».
- Форсировка пути на значение «ложь» предотвращает вход ПФС в этот путь и выполнение его шагов.
- Когда вы удалите или запретите форсировку, ПФС сможет выполнить шаги данного пути.

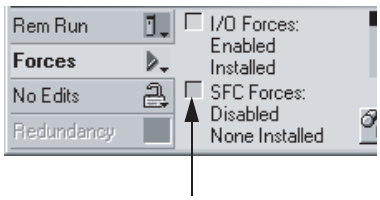
Добавить форсировку ПФС

Для замены логики ПФС используйте форсировку ПФС:

ВНИМАНИЕ

Форсировка может привести к неожиданному движению механизмов, что может повлечь за собой травмы персонала. Перед использованием форсировки определите, как она повлияет на ваш механизм или процесс, и удалите людей из зоны действия механизмов.

- Разрешение форсировки ПФС вызывает переход механизма или процесса в другое состояние или фазу.
- Если форсировка разрешена, она начинает действовать немедленно.



1. Каково состояние индикатора форсировки ПФС?

Если:	Тогда:
не горит	В данный момент не существует форсировок ПФС.
мигает	Форсировки ПФС не активны. Но по крайней мере одна форсировка существует в вашем проекте. Когда вы разрешиете форсировки ПФС, начнут действовать все форсировки ПФС.
горит не мигая	Форсировки ПФС разрешены (активны). Когда вы устанавливаете (добавляете) форсировку, она начинает действовать немедленно.

2. Откройте процедуру ПФС (SFC).
3. Щелкните правой клавишей мыши на переходе или начале параллельного пути, который вы хотите форсировать, и выберите *Force TRUE* (только для переходов) или *Force FALSE*.
4. Разрешены ли форсировки ПФС? (См. шаг1.)

Если:	То:
нет	В меню *Logic(Логика)* выберите *SFC Forcing >Enable All SFCForces*. Затем выберите *Yes* для подтверждения.
да	Стоп.

Удаление или запрещение форсировок

ВНИМАНИЕ

Изменение форсировок может привести к неожиданному движению механизмов, что может повлечь за собой травмы персонала. Перед использованием форсировки определите, как изменение повлияет на ваш механизм или процесс, и удалите людей из зоны действия механизмов.

Если вы хотите:	И:	То:
Остановить отдельную форсировку	Сохранить другие форсировки разрешенными и действующими	Удалите отдельную форсировку
Остановить все форсировки ввода/вывода, но оставить все форсировки ПФС активными	Сохранить все форсировки ввода/вывода в вашем проекте	Запретите все форсировки ввода/вывода
	Удалить все форсировки ввода/вывода в вашем проекте	Удалите все форсировки ввода/вывода
Остановить все форсировки ПФС, но оставить все форсировки ввода/вывода активными	Сохранить все форсировки ПФС в вашем проекте	Запретите все форсировки ПФС
	Удалить все форсировки ПФС в вашем проекте	Удалите все форсировки ПФС

Удаление отдельной форсировки

ВНИМАНИЕ

Если вы удаляете отдельную форсировку, другие форсировки остаются в состоянии «разрешены» и новая форсировка начинает действовать немедленно.

Перед удалением форсировки определите, как это изменение повлияет на ваш механизм или процесс, и удалите людей из зоны действия механизмов.

1. Откройте процедуру, которая содержит форсировку, которую вы собираетесь удалить.
2. На каком языке сделана данная процедура?

Если:	То:
ПФС	Переходите к шагу 4
Релейной логики	Переходите к шагу 4
Функциональных блоков	Переходите к шагу 3
Структурированного текста	Переходите к шагу 3

3. Щелкните правой клавишей мыши на теге, который имеет форсировку и выберите *Monitor...* Если необходимо, то раскройте этот тег, чтобы видеть значение, которое было форсировано (например значение BOOL тега DINT).
4. Щелкните правой клавишей мыши на теге или элементе, который имеет форсировку и выберите *Remove Force (Удалить форсировку)*.

Запрещение всех форсировок ввода/вывода

В меню *Logic* выберите *I/O Forcing >Disable All I/O Forces*. Затем выберите *Yes* для подтверждения.

Удаление всех форсировок ввода/вывода

В меню *Logic* выберите *I/O Forcing > Remove All I/O Forces*. Затем выберите *Yes* для подтверждения.

Запрещение всех форсировок ПФС

В меню *Logic* выберите *SFC Forcing > Disable All SFC Forces*. Затем выберите *Yes* для подтверждения.

Удаление всех форсировок ПФС

В меню *Logic* выберите *SFC Forcing > Remove All SFC Forces*. Затем выберите *Yes* для подтверждения.

Обработка основной ошибки

Использование этой главы

Используйте эту главу для построения логики обработки определенных ошибок.

Создание процедуры обработки ошибок

Если возникает ошибка, достаточно серьезная для прекращения работы контроллера, контроллер выдает основную ошибку и прекращает выполнение логической части.

- В зависимости от Вашего приложения, Вы можете не захотеть, чтобы основная ошибка прекращала работу всей системы.
- В этой ситуации Вы можете использовать процедуру обработки ошибок для исправления конкретной ошибки и разрешить по крайней мере некоторой части Вашей системы продолжать выполнение.

ПРИМЕР

Использование процедуры обработки ошибок

В системе, которая использует такой способ нумерации рецептов, как косвенные адреса, неправильно введенный номер может вызвать основную ошибку, такую как ошибка типа 4 с кодом 20.

Чтобы избежать выключения всей системы, процедура обработки ошибок исправляет все основные ошибки типа 4 с кодом 20.

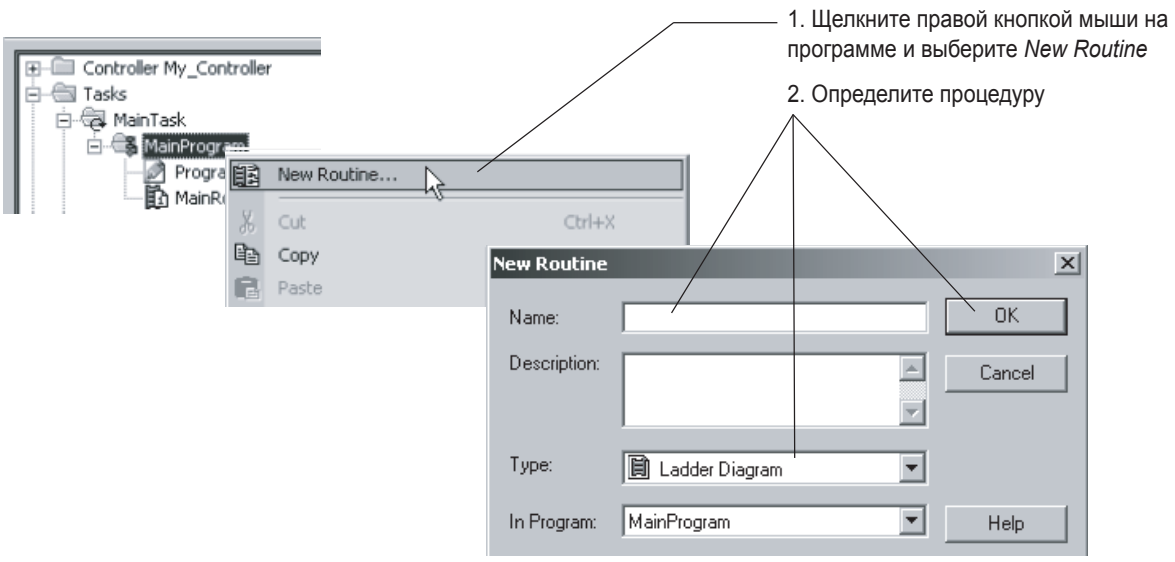
Выбор местоположения процедуры обработки ошибок

Процедура обработки ошибок позволяет логике Вашей программы предпринимать определенные действия после ошибки, например, исправление ошибки и продолжение выполнения. Где расположить процедуру, зависит от типа ошибки, которую нужно обрабатывать:

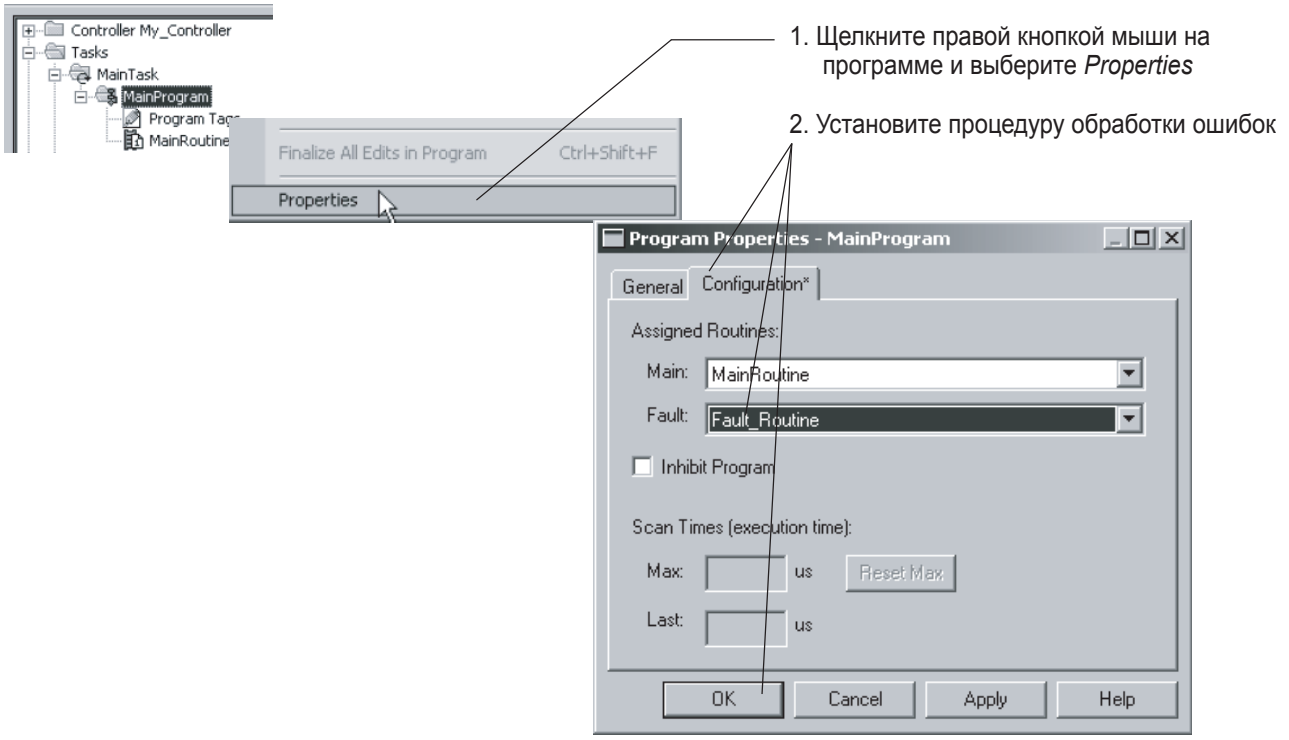
Если Вы хотите предпринять действия по исправлению ошибки когда:		**Сделайте следующее:**	**Смотри страницу:**
Состояние:	**Тип ошибки:**		
Ошибка при выполнении инструкции	4	Создайте процедуру обработки ошибок для программы	15-2
Сбой связи с модулем I/O	3	Создайте процедуру для обработчека ошибок контроллера.	15-3
Истекло время сторожевого таймера задачи	6		
Во время загрузки проекта в контроллер переключатель находится в положении RUN	8		
Ошибка оси перемещения	11		
Питание контроллера включается в режиме выполнения/удаленного выполнения	1	Создайте процедуру для обработчика включения питания	15-4

Создание процедуры обработки ошибок для программы

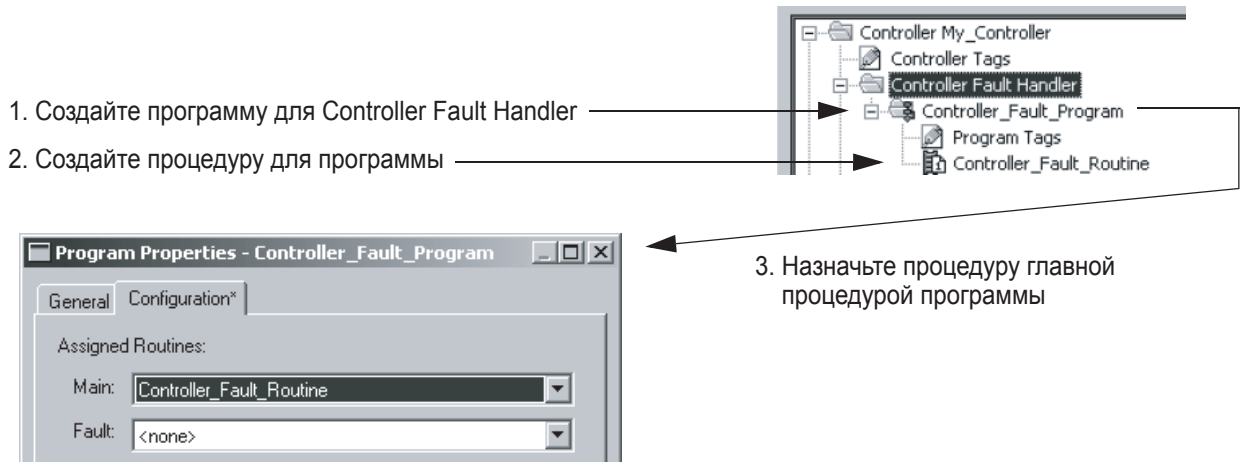
Создание процедуры



Назначение процедуры процедурой обработки ошибок



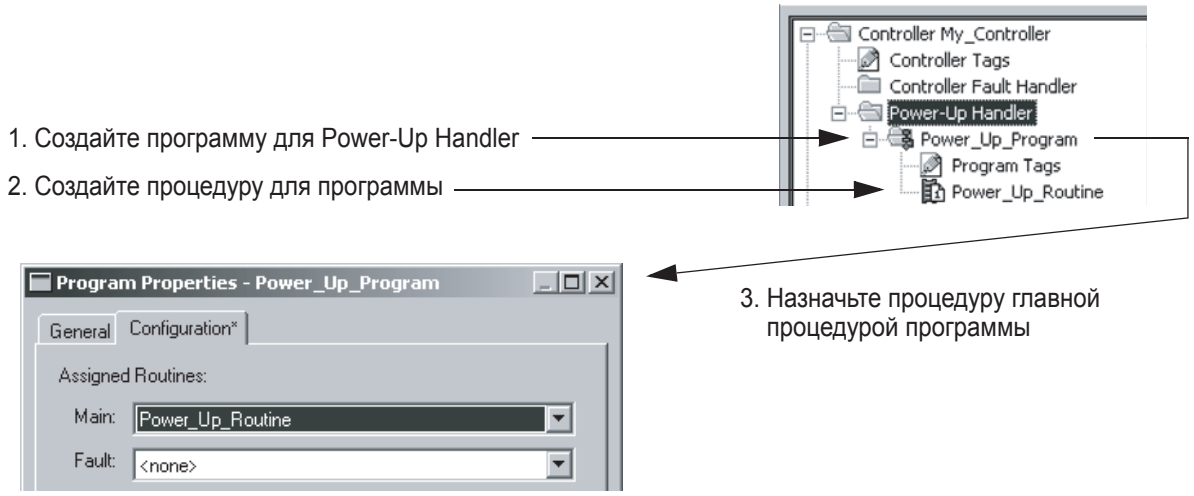
Создание процедуры для обработки ошибок контроллера



Создание процедуры для обработчика включения питания

Обработчик включения питания - это дополнительная задача, которая выполняется когда включается питание контроллера в режиме выполнения/удаленного выполнения. . Используйте обработчик включения питания , когда Вы хотите добиться чего-либо из перечисленного в нижеследующей таблице после потери и последующего восстановления питания.

Чтобы:	Сделайте следующее:
Предотвратить возвращение контроллера в режим выполнения/удаленного выполнения	Оставьте пустой процедуру для обработчика включения питания (Power-Up Handler). Когда питание восстановлено, возникает основная ошибка (тип1, код 1) и контроллер переходит в режим Faulted.
Вернуться к нормальной работе и предпринять определенные действия, когда питание восстановлено	В процедуре для обработчика включения питания: 1. Сбросьте основную ошибку (тип 1, код 1). 2. Введите логику действий.



Программный сброс основной ошибки

Чтобы сбросить основную ошибку, которая возникла во время выполнения Вашего проекта, проделайте следующие действия в соответствующей процедуре (Смотри *"Выбор местоположения процедуры обработки ошибок"* на странице 15-2.)

Шаг:	Страница:
☐ Создайте тип данных для хранения информации об ошибке	15-5
☐ Определите тип и код ошибки	15-6
☐ Проверьте конкретную ошибку	15-7
☐ Сбросьте ошибку	15-7

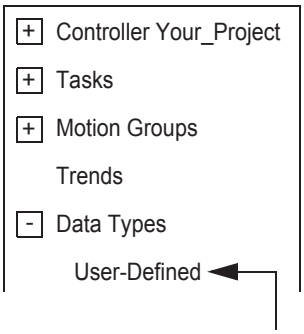
Создание типа данных для хранения информации об ошибке

Контроллеры Logix5000 хранят системную информацию в объектах. В отличие от контроллеров PLC-5 и SLC500, у них нет файла состояния.

- Для доступа к системной информации Вы используете инструкции Get System Value (GSV) или Set System Value (SSV).
- Для доступа к информации о состоянии программы Вы обращаетесь к объекту PROGRAM.
- Для доступа к информации об ошибке Вы обращаетесь к следующему атрибуту объекта PROGRAM.

Атрибут:	Тип данных:	Инструкция:	Описание:
MajorFaultRecord	DINT[11]	GSV SSV	Записывает основные ошибки для программы. Задайте имя программы для определения нужного объекта PROGRAM. (Или определите THIS для доступа к объекту PROGRAM для программы, которая содержит инструкцию GSV или SSV.)

Чтобы создать новый тип данных:

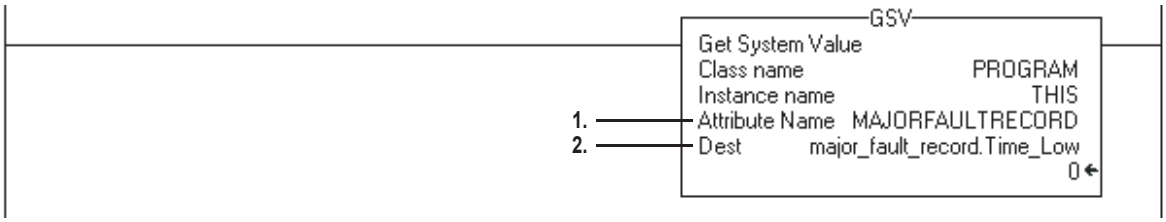


Щелкните правой клавишей мыши и выберите *New Data Type (Новый тип данных).*

Для упрощения доступа к атрибуту MajorFaultRecord, создайте следующий определяемый пользователем тип данных:

Тип данных: FAULTRECORD				
Имя	FAULTRECORD			
Описание	Хранит атрибут MajorFaultRecord или атрибут MinorFaultRecord объекта PROGRAM			
Члены				
	Имя	**Тип данных**	**Формат**	**Описание**
	Time_Low	STRING	Десятичный	Нижние 32 бита значения временной метки ошибки
	Time_High	DINT	Десятичный	Верхние 32 бита значения временной метки ошибки
	Type	INT	Десятичный	Тип ошибки (программная, I/O и др.)
	Code	INT	Десятичный	Уникальный код ошибки
	Info	DINT[8]	Шестнадца-теричный	Характерная информация об ошибке

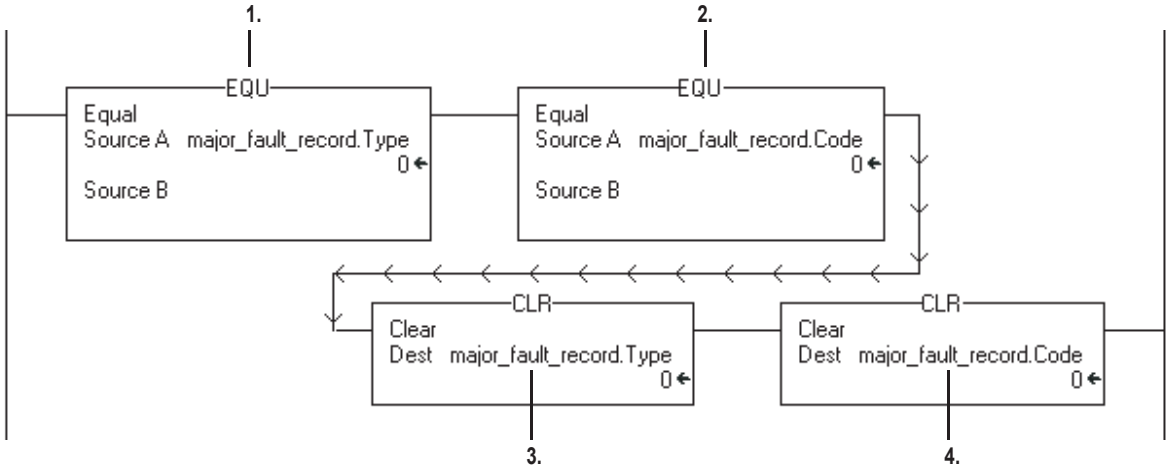
Определение типа и кода ошибки



1. Инструкция GSV имеет доступ к атрибуту MAJORFAULTRECORD этой программы. В этом атрибуте хранится информация об ошибке.
2. Инструкция GSV хранит информацию об ошибке в теге major_fault_record. Когда Вы вводите тег, который основан на структуре, введите первый член этого тега.

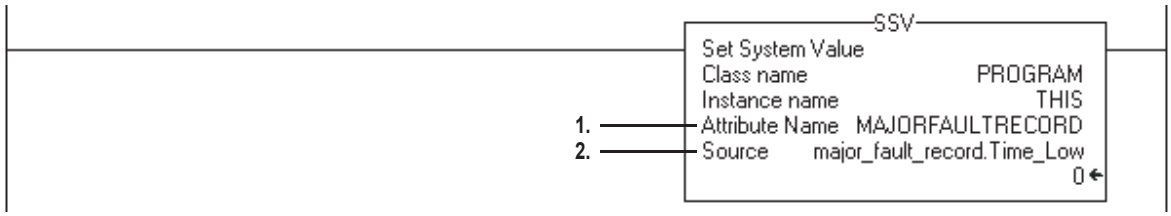
Имя тега	**Тип**
major_fault_record	FAULTRECORD

Проверка ошибки



1. Инструкция EQU проверяет определенный тип ошибки, например, программная, I/O. В Source B введите значение типа ошибки, которую вы хотите сбросить.
2. Инструкция EQU проверяет код ошибки. В Source B введите значение кода ошибки, которую Вы хотите сбросить.
3. Инструкция CLR устанавливает значение типа ошибки в теге *major_fault_record* равным нулю.
4. Инструкция CLR устанавливает значение кода ошибки в теге *major_fault_record* равным нулю.

Сброс ошибки



1. Инструкция SSV записывает новые значения в атрибут MAJORFAULTRECORD программы.
2. Инструкция SSV записывает значения, содержащиеся в теге *major_fault_record.* С того момента, как члены Type и Code установлены равными нулю, ошибка сброшена и контроллер продолжает выполнение.

Сброс основной ошибки во время предварительного сканирования

Если контроллер выдает ошибку сразу после того, как Вы переключили его в режим выполнения (Run), проверьте операцию **предварительного сканирования** на ошибку. В зависимости от версии Вашего контроллера, выход индекса массива за заданную размерность массива во время предварительного сканирования может вызывать, а может и не вызывать генерацию ошибки:

Если Ваш контроллер версии:	Тогда:
11.x или более ранний	Во время предварительного сканирования индекс массива, находящийся за пределами размерности массива вызывает основную ошибку.
12.x	Смотри документацию к программно-аппаратным средствам контроллера.
13.0 или более поздний	Во время предварительного сканирования контроллер автоматически сбрасывает все ошибки связанные с выходом индексов за пределы размерности массива.

Чтобы сбросить основную ошибку, возникшую во время предварительного сканирования:

☐ Определите, когда контроллер осуществляет предварительное сканирование

☐ Определите тип и код ошибки

☐ Проверьте конкретную ошибку

☐ Сбросьте ошибку

Определение, когда контроллер в состоянии предварительного сканирования

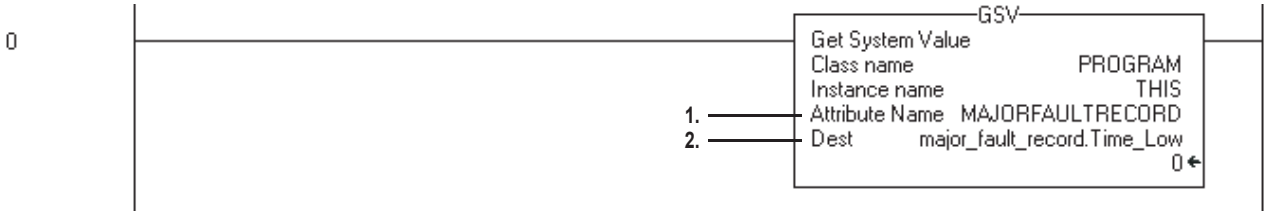
В главную процедуру Вашей программы введите следующую цепочку:

1. Введите эту цепочку как первую цепочку в главной процедуре программы
2. Процедура обработки ошибок этой программы использует состояние этого бита для определения, возникла эта ошибка во время предварительного сканирования или во время нормального сканирования логики:
   - Во время предварительного сканирования этот бит выключен. (Во время предварительного сканирования контроллер сбрасывает все биты, на которые ссылаются инструкции OTE.)
   - Как только контроллер начнет выполнение логики, этот бит будет все время включен.

Имя тега	Тип
CPU_scanning	BOOL

Определение типа и кода ошибки

Введите эту цепочку в процедуру обработки ошибок программы:

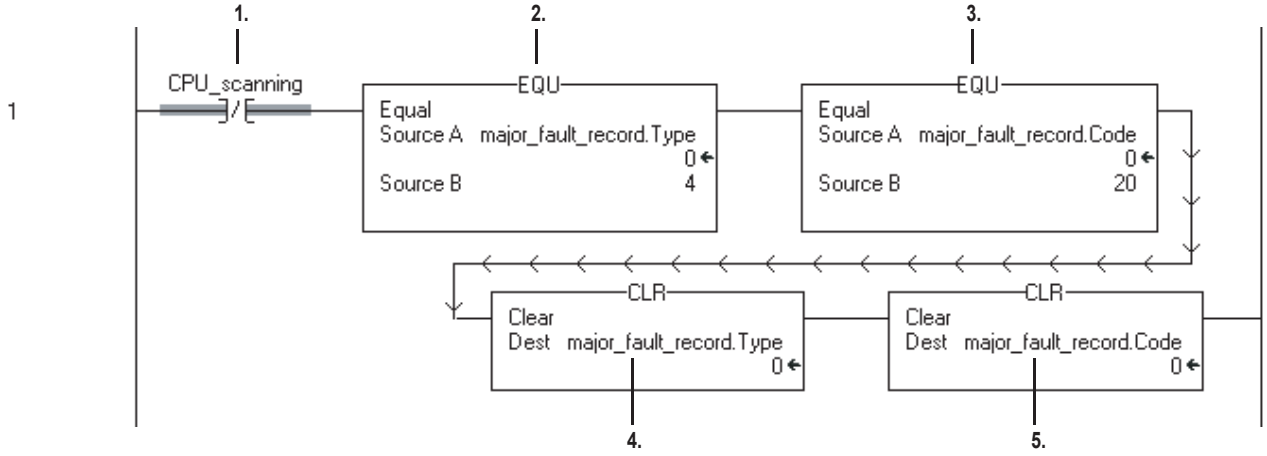


1. Инструкция GSV имеет доступ к атрибуту MAJORFAULTRECORD программы. Атрибут хранит информацию об ошибке.
2. Инструкция GSV хранит информацию об ошибке в теге major_fault_record. Когда Вы вводите тег, основанный на структуре, введите первый член тега.

Имя тега	Тип
major_fault_record	FAULTRECORD

Проверка ошибки:

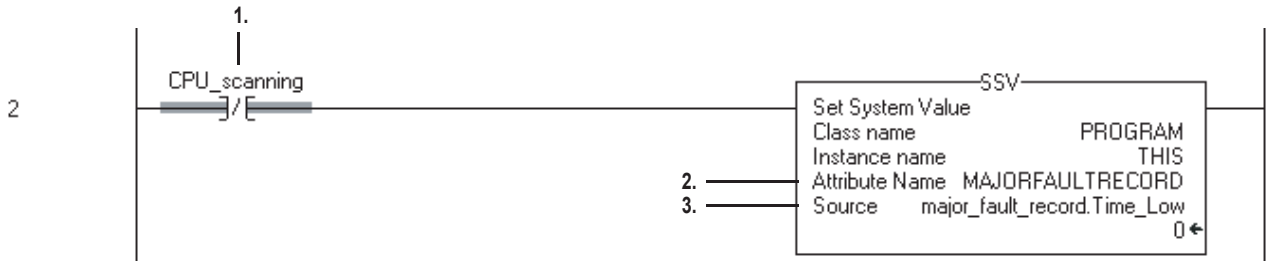
Введите эту цепочку в процедуру обработки ошибок программы:



1. Во время предварительного сканирования биты всех инструкций OTE выключены и эта инструкция имеет значение “истина”. Как только контроллер начинает выполнение логики, эта инструкция принимает значение “ложь”.
2. Эта инструкция EQU проверяет ошибку типа 4, которая означает, что инструкция в этой программе вызвала ошибку.
3. Инструкция EQU проверяет ошибку с кодом 20, которая означает или что индекс массива слишком большой, или что значение POS или LEN структуры CONTROL неверное.
4. Инструкция CLR устанавливает в ноль значение типа ошибки в теге *major_fault_record*.
5. Инструкция CLR устанавливает в ноль значение кода ошибки в теге *major_fault_record*.

Сброс ошибки

Введите эту цепочку в процедуру обработки ошибок программы:

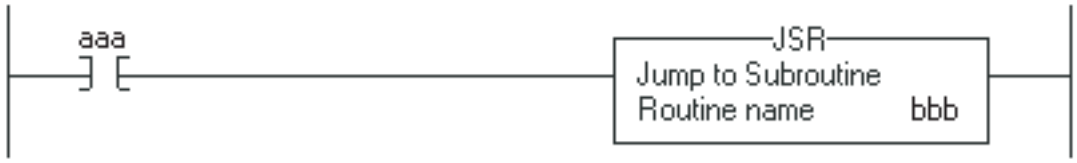


1. Во время предварительного сканирования биты всех OTE инструкций выключены и инструкция имеет значение “истина”. Как только контроллер начинает выполнять логическую часть, эта инструкция принимает значение “ложь”.
2. Инструкция SSV записывает новые значения в атрибут MAJORFAULTRECORD программы.
3. Инструкция SSV записывает значения, содержащиеся в теге *major_fault_record.* Как только члены *Type* и *Code* устанавливаются в ноль, ошибка сбрасывается и контроллер продолжает выполнение.

Тестирование процедуры обработки ошибок

Вы можете использовать инструкцию JSR для тестирования процедуры обработки ошибок программы без создания ошибки (т.е. имитация ошибки):

1. Создайте тег BOOL, который будет использоваться для инициализации ошибки.
2. В главной процедуре или подпрограмме программы введите следующую цепочку:

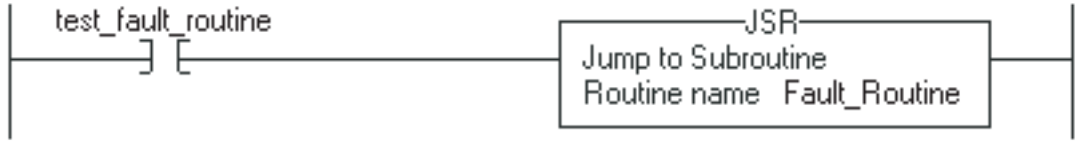


Где:	Это:
aaa	тег, который будет использоваться для инициализации ошибки (шаг 1)
bbb	Процедура обработки ошибок программы

3. Чтобы имитировать ошибку, установите входное состояние.

ПРИМЕР Тестирование процедуры обработки ошибок

Когда *test_fault_routine* включен, возникает основная ошибка и контроллер начинает выполнение *Fault_Routine*.



Создание основной ошибки, определяемой пользователем

Если Вы хотите приостановить (отключить) контроллер на основании условий в Вашем приложении, создайте определяемую пользователем основную ошибку:

- Тип ошибки = 4.
- Определите значение кода ошибки. Выберите значение в диапазоне от 990 до 999. Эти коды зарезервированы для определяемых пользователем ошибок.
- Контроллер обрабатывает эту ошибку также, как и другие основные ошибки:
  - Контроллер переходит в **режим ошибки** (основной ошибки) и прекращает выполнение алгоритма.
  - Выходные данные установливаются в их сконфигурированное состояние или приобретают значение, заданное для режима ошибки.

ПРИМЕР

Определяемая пользователем основная ошибка

Когда Tag_1.0 = 1, выдайте основную ошибку и сгенерируйте код ошибки 999.

Чтобы создать определяемую пользователем основную ошибку:

☐ Создайте процедуру обработки ошибки для программы

☐ Сконфигурируйте программу для использования процедуры обработки ошибки

☐ Перейдите к процедуре обработки ошибки

Создание процедуры обработки ошибки для программы

Имеется ли уже процедура обработки ошибки в программе?

Если:	Тогда:
Да	См «Переход к процедуре обработки ошибки» на стр. 15-14
Нет	Создайте процедуру обработки ошибки:

1. В организаторе контроллера щелкните правой кнопкой мыши на программе и выберите *New Routine*.
2. В поле имени напечатайте имя процедуры обработки ошибки (`name_of_fault_routine`).
3. Из выпадающего списка *Type* выберите *Ladder*.
4. Нажмите OK.

Конфигурирование программы для использования процедуры обработки ошибки

1. В организаторе контроллера щелкните правой кнопкой мыши на программе и выберите New Routine.
2. Выберите закладку Configuration.
3. Из выпадающего списка Fault выберите fault routine.
4. Нажмите OK.

Переход к процедуре обработки ошибки

В главную процедуру программы введите следующую цепочку:

где:	это:
Fault_Routine	Имя процедуры обработки ошибки в программе
x	Значение кода ошибки

ПРИМЕР Создание определяемой пользователем основной ошибки

Когда *Tag_1.0 = 1*, выполнение переходит к *name_of_fault_routine*. Возникает основная ошибка и контроллер переходит в состояние ошибки. Выходные данные переходят в состояние ошибки. В диалоговом окне Controller Properties, на закладке Major Faults показывается код 999.

Коды основных ошибок

Используйте нижеследующую таблицу для определения причины и принятия мер по устранению основной ошибки. Тип и код соответствуют типу и коду, отображенным в:

- диалоговом окне Controller Properties на закладке Major Faults
- в атрибуте MAJORFAULTRECORD объекта PROGRAM

Таблица 15.1 Типы и коды основных ошибок

Тип	Код	Причина	Метод устранения
1	1	Контроллер находится в режиме выполнения (Run).	Запустите программу обработки потери питания.
1	60	Для контроллера, на котором не установлена карта Compact Flash: • обнаружена неисправимая ошибка • проект стерт из памяти	1. Сбросьте ошибку. 2. Загрузите проект. 3. Перейдите в режим удаленного выполнения/выполнения (remote run/ run). Если проблема остается: 1. Перед тем как вы выключите и снова включите компьютер, запишите состояние светодиодных индикаторов OK и RS232. 2. Свяжитесь со службой поддержки Rockwell. Смотри в конце данной публикации.
1	61	Для контроллера с установленной картой Compact Flash, контроллер: • обнаружена неисправимая ошибка • на карту Compact Flash записана диагностическая информация • проект стерт из памяти	1. Исправьте ошибку. 2. Загрузите проект. 3. Перейдите в режим удаленного выполнения/выполнения. Если проблема остается, свяжитесь со службой поддержки Rockwell. Смотри в конце данной публикации.
3	16	Необходимое соединение I/O модуля разорвано.	Проверьте модуль I/O в шасси. Проверьте требования по электронному ключу. Посмотрите в свойствах контроллера закладку Major Fault и в свойствах модуля закладку Connections для получения дополнительной информации об ошибке.
3	20	Проблема с шасси ControlBus.	Невозможно исправить – замените шасси.
3	23	По крайней мере одно необходимое соединение не установлено перед переходом в режим выполнения Run.	Дождитесь, пока индикатор I/O контроллера не станет зеленым перед тем, как переходить в режим Run.
4	16	Встретилась неизвестная инструкция.	Удалите неизвестную инструкцию. Это могло произойти в процессе перекодирования программы.
4	20	Индекс массива слишком велик, управляющая структура .POS или .LEN неверна.	Присвойте значения из правильного диапазона. Не выходите за пределы размера массива и не превышайте определенной размерности.
4	21	Управляющая структура .POS или .LEN < 0.	Задайте значения > 0.
4	31	Параметры инструкции JSR не соответствуют параметрам инструкции SBR или RET	Передавайте подходящее количество параметров. Если передано слишком много параметров, лишние игнорируются без ошибки.
4	34	Инструкция таймера имеет отрицательное заранее установленное или накопленное значение.	Настройте программу так ,чтобы она не загружала отрицательное заранее установленное или накопленное значение в таймер.
4	42	Переход (JMP) на удаленную или несуществующую метку.	Исправьте указание на метку в JMP или добавьте отсутствующую метку.

Таблица 15.1 Типы и коды основных ошибок (продолжение)

Тип	Код	Причина	Метод устранения
4	82	ПФС обратилась к подпрограмме, а подпрограмма попыталась обратиться к ПФС. Происходит, когда ПФС использует JSR или FOR инструкцию для обращения к подпрограмме.	Удалите переход к вызывающей ПФС.
4	83	Тестируемые данные оказались вне требуемых пределов.	Измените значение, чтобы оно было в требуемых пределах.
4	84	Переполнение стека.	Сократите уровень вложенности подпрограммы или число передаваемых параметров.
4	89	В инструкции SFR целевая процедура не содержит целевой ступени.	Исправьте целевой объект SFR или добавьте пропущенную ступень.
6	1	Истекло время ожидания выполнения задания. Задача пользователя не завершена в заданный период времени. Ошибка в программе вызвала зацикливание или нельзя выполнить задачу так быстро, как указано, или задача с более высоким приоритетом удерживает данную задачу от завершения.	Увеличьте допустимое время выполнения задачи, сократите время выполнения, увеличьте приоритет данной задачи, упростите задачи с более высоким приоритетом или переместите часть кода на другой контроллер.
7	40	Попытка записи в энергонезависимую память не удалась.	1. Попробуйте еще раз записать проект в энергонезависимую память. 2. Если попытка записать проект в энергонезависимую память не удалась, замените плату запоминающего устройства.
7	42	Попытка загрузки из энергонезависимой памяти не удалась, т.к. версия встроенного программного обеспечения проекта в энергонезависимой памяти не согласовывается с версией встроенного программного обеспечения контроллера.	Обновите встроенные программные средства контроллера до версии встроенных программных средств проекта в энергонезависимой памяти.
8	1	Не удается перевести контроллер в режим выполнения с помощью переключателя в процессе загрузки.	Подождите окончания загрузки и сбросьте ошибку.
11	1	Фактическое положение вышло за пределы положительного предела перебега.	Переместите ось в отрицательном направлении до позиции, когда она будет в пределах перебега и затем выполните Motion Axis Fault Reset.
11	2	Фактическое положение вышло за пределы отрицательного предела перебега.	Переместите ось в положительном направлении до позиции, когда она будет в пределах перебега и затем выполните Motion Axis Fault Reset.
11	3	Фактическое положение вышло за пределы допустимого положения.	Переместите положение в пределы допустимого положения и затем выполните Motion Axis Fault Reset.
11	4	Разорвано соединение с каналом кодировщика A,B или Z.	Восстановите соединение с каналом кодировщика и затем выполните Motion Axis Fault Reset.
11	5	Обнаружены помехи кодировщика или сигналы кодировщика не в квадратурах.	Исправьте кабельное соединение с кодировщиком и затем выполните Motion Axis Fault Reset.
11	6	Активирован входной сигнал Drive Fault (Ошибка привода).	Сбросьте Drive Fault и затем выполните Motion Axis Fault Reset.
11	7	Синхронное соединение неисправно.	Сначала выполните Motion Axis Fault Reset. Если не заработает, вытащите и вставьте блок сервопривода. Если ошибка не исправлена, замените блок сервопривода.

Таблица 15.1 Типы и коды основных ошибок (продолжение)

Тип	Код	Причина	Метод устранения
11	8	Блок сервопривода обнаружил серьезный аппаратный отказ.	Замените блок.
11	9	Асинхронное соединение неисправно.	Сначала выполните Motion Axis Fault Reset. Если это не помогло, вытащите и вставьте блок сервопривода. Если ошибка не исправлена, замените блок сервопривода.
11	32	При выполнении задачи перемещения произошло перекрытие.	Частота обновления хода группы слишком высока для поддержания правильности операции. Очистите тег ошибки группы, повысьте частоту обновления и затем сбросьте основную ошибку.

Для заметок:

Отслеживание неосновных ошибок

Когда использовать эту процедуру

Если возникает некритическая ошибка, которая не приводит к отключению контроллера, контроллер выдает неосновную ошибку.

- Контроллер продолжает выполнение.
- Вам не нужно сбрасывать неосновную ошибку.
- Чтобы оптимизировать время выполнения и гарантировать правильность выполнения программы, нужно отслеживать и исправлять неосновные ошибки.

Отслеживание неосновных ошибок

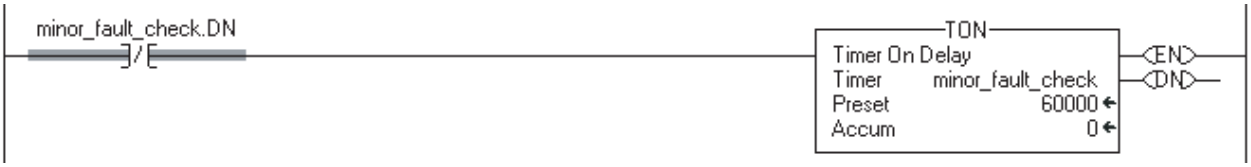
Использование релейной логики для получения информации о неосновной ошибке:

<table>
<tr><th>Чтобы проверить:</th><th>Сделайте следующее:</th></tr>
<tr><td>Перекрытия периодических задач</td><td>1. Введите GSV инструкции, которые получают атрибут MinorFaultBits объекта FAULTLOG.
2. Проверьте разряд 6.</td></tr>
<tr><td>Загрузку энергонезависимой памяти</td><td>1. Введите GSV инструкции, которые получают атрибут MinorFaultBits объекта FAULTLOG.
2. Проверьте разряд 7.</td></tr>
<tr><td>Проблемы с последовательным портом</td><td>1. Введите GSV инструкции, которые получают атрибут MinorFaultBits объекта FAULTLOG.
2. Проверьте разряд 9.</td></tr>
<tr><td>Разрядку батареи</td><td>1. Введите GSV инструкции, которые получают атрибут MinorFaultBits объекта FAULTLOG.
2. Проверьте разряд 10.</td></tr>
<tr><td>Проблемы с инструкциями</td><td>1. Создайте определяемый пользователем тип данных, который будет хранить информацию об ошибке. Назовите этот тип данных FaultRecord и определите следующие атрибуты:
<table>
<tr><th>Имя</th><th>Тип данных</th><th>Формат</th></tr>
<tr><td>TimeLow</td><td>DINT</td><td>десятичный</td></tr>
<tr><td>TimeHigh</td><td>DINT</td><td>десятичный</td></tr>
<tr><td>Type</td><td>INT</td><td>десятичный</td></tr>
<tr><td>Code</td><td>INT</td><td>десятичный</td></tr>
<tr><td>Info</td><td>DINT[8]</td><td>шестнадцатеричный</td></tr>
</table>
2. Создайте тег, в котором будут храниться значения атрибута MinorFaultRecord. Тип данных выбирается из шага 1.
3. Проверьте S:MINOR.
4. Если S:MINOR включен, используйте инструкцию GSV для получения значений атрибута MinorFaultRecord
5. Если Вы хотите обнаружить неосновную ошибку, которая вызвана другой инструкцией, сбросьте S:MINOR (S:MINOR остается установленным до конца сканирования).</td></tr>
</table>

Следующий пример показывает проверку наличия предупреждения о разрядке батареи.

ПРИМЕР Проверка неосновной ошибки.

Minor_fault_check функционирует 1 минуту (60000 мс) и затем автоматически перезагружается.



Каждую минуту *minor_fault_check.DN* включается для одного сканирования. Когда это происходит, инструкция GSV получает значение атрибута *MinorFaultBits* объекта *FAULTLOG*, и записывает его в тег *minor_fault_bit*. Так как инструкция GSV выполняется только 1 раз за минуту, время сканирования большинства сканирований уменьшается.

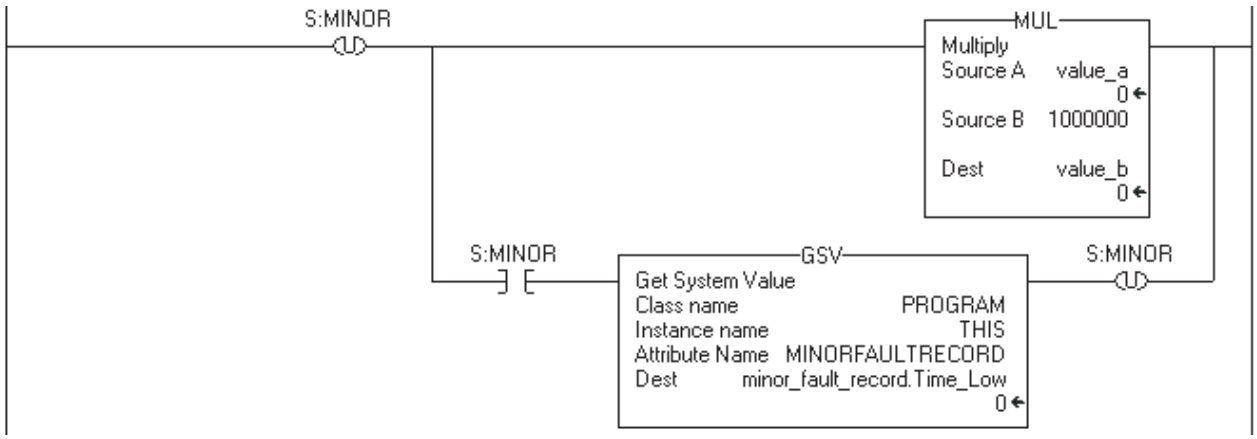
Если minor_fault_bits.10 включено, то батарея разряжена.

Следующий пример показывает проверку неосновной ошибки, вызванной конкретной инструкцией.

ПРИМЕР Проверка неосновной ошибки, вызванной инструкцией.

Value_a умножается на 1000000 и проверяется неосновная ошибка, связанная с математическим переполнением:

- Чтобы быть уверенным , что предыдущая инструкция не вызвала ошибку, сначала цепочка очищает S:MINOR.
- Затем выполняется инструкция умножения.
- Если инструкция вызывает неосновную ошибку, контроллер устанавливает S:MINOR.
- Если S:MINOR установлен, инструкция GSV получает информацию об ошибке и сбрасывает S:MINOR.



Коды неосновных ошибок

Используйте следующую таблицу для определения причины и способов исправления неосновных ошибок. Тип и код соответствуют типу и коду, отображаемому в:

- диалоговом окне свойств контроллера на закладке minor faults
- атрибуте MINORFAULTRECORD объекта PROGRAM

Таблица16.1 Типы и коды неосновных ошибок

Тип	Код	Причина	Метод исправления
4	4	Возникновение арифметического переполнения в инструкции.	Исправьте программу, проверив арифметические операции (порядок) или задаваемые значения.
4	5	В инструкции GSV/SSV определенный экземпляр не найден.	Проверьте имя экземпляра.
4	6	В инструкции GSV/SSV • не поддерживается определенное имя класса • определенное имя атрибута не действительно	Проверьте имя класса и атрибута.
4	7	Тег назначения GSV/SSV слишком мал для хранения всей информации.	Сделайте этот тег достаточно большим.
4	35	Промежуток времени PID < 0	Присвойте промежутку времени значение PID большее 0.
4	36	Заданное значение PID выходит из диапазона.	Присвойте такое значение, которое не выходит из диапазона.
4	51	Значение LEN строкового тега больше, чем размер DATA этого строкового тега.	1. Проверьте, чтобы инструкция не производила запись в компонент LEN строкового тега. 2. В LEN введите количество о символов, содержащихся в строке.
4	52	Выходная строка больше получателя.	Создайте новый строковый тип данных, достаточно большой для хранения выходной строки. Используйте новый строковый тип данных как место назначения для выходной строки.
4	53	Выходное число выходит за пределы типа данных адресата.	• Сократите размер ASCII значения • Или используйте больший тип данных для адресата
4	56	Значение Start или Quantity неверное.	1. Проверьте, что значение Start находится в диапазоне от 1 до размера DATA источника. 2. Проверьте, чтобы сумма Start и Quantity была меньше или равна размера DATA источника.
4	57	Не удалось выполнить инструкцию AHL, т.к. для последовательного порта не установлено подтверждение связи.	• Измените настройку Control Line последовательного порта. • Или удалите AHL инструкцию.
6	2	Перекрытия периодических задач.Периодическая задача не закончена до начала ее нового выполнения.	Упростите программу (программы) или удлините период, или поднимите относительный приоритет и т.д.
7	49	Проект загружен из энергонезависимой памяти.	

Таблица16.1 Типы и коды неосновных ошибок (продолжение)

Тип	Код	Причина	Метод исправления
9	0	Неизвестная ошибка во время обслуживания последовательного порта.	Обратитесь к персоналу, оказывающему техническую поддержку (GTS).
9	1	Линия CTS неверна для текущей конфигурации.	Отключите и подключите заново кабель последовательного порта к контроллеру. Убедитесь, что кабель подключен правильно.
9	2	Ошибка списка опросов. Была обнаружена следующая проблема с основным списком опросов DF1: установлено больше станций, чем размер файла, установлено более 255 станций, произведена попытка индексации после конца списка или произведен опрос широковещательных адресов (STN #255).	Проверьте следующие ошибке в списке опросов: • Общее число станций больше, чем пространство в теге списка опросов. • Общее число станций больше 255. • Текущий указатель на станцию больше чем конечное значение тега списка опросов. • Встретился номер станции больший 254.
9	5	Истекло время ожидания подчиненного опроса DF1 Истекло время ожидания системы безопасности опросов для подчиненного устройства. Ведущее устройство не обратилось к контроллеру за определенный период времени.	Установите и исправьте задержку опроса.
9	9	Потерян контакт с модемом. Линии управления DSD и/или DSR не получены в некорректной последовательности и/или состоянии.	Исправьте модемное соединение с контроллером.
10	10	Батарея не обнаружена или должна быть заменена.	Установите новую батарею.

Для заметок:

Сохранение и загрузка проекта с помощью энергонезависимой памяти

Когда использовать эту процедуру

.

ВАЖНО

Энергонезависимая память сохраняет содержимое пользовательской памяти при сохранении проекта.

- Изменения, которые вы вносите после сохранения проекта, *не* отражаются в энергонезависимой памяти.
- Если вы вносите изменения в проект, но не сохраняете эти изменения, вы затрете их при загрузке проекта из энергонезависимой памяти. В этом случае вам придется выгрузить или загрузить проект, чтобы перейти в режим онлайн.
- Если вы хотите сохранить такие изменения, как правки, внесенные в режиме онлайн, значения тегов или сетевое расписание ControlNet, сохраните проект снова после внесения изменений.

Используйте эту процедуру для **сохранения** или **загрузки** проекта с помощью **энергонезависимой памяти** контроллера.

- Если происходит потеря питания контроллера при отсутствии аккумуляторной батареи достаточной мощности, то проект в пользовательской памяти контроллера утрачивается.
- Энергонезависимая память позволяет вам сохранять копию вашего проекта в контроллере. Контроллеру не требуется питание для сохранения этой копии.
- Вы можете загружать копию проекта из энергонезависимой памяти в пользовательскую память контроллера:
  - при каждом включении питания
  - всякий раз, когда включается питание контроллера при отсутствии в нем проекта
  - в любой момент через программное обеспечение RSLogix 5000.

Как использовать эту процедуру

Если вы хотите:	Смотрите:
ознакомиться с предварительной информацией о том, как использовать энергонезависимую память	«Прежде чем использовать энергонезависимую память», стр. 17-2
сохранить проект в энергонезависимой памяти контроллера	«Сохранение проекта», стр. 17-9
заменить текущий проект в контроллере проектом, хранящимся в энергонезависимой памяти контроллера	«Загрузка проекта», стр. 17-12
загрузить проект после очистки памяти в результате потери питания при отсутствии аккумуляторной батареи	
использовать релейную логику, чтобы пометить, что проект загружен из энергонезависимой памяти	«Пометка для загрузки», стр. 17-14
удалить проект из энергонезависимой памяти контроллера	«Очистка энергонезависимой памяти», стр. 17-15
• задать другой проект для загрузки с карты CompactFlash • изменить параметры загрузки для проекта на карте CompactFlash	«Использование считывателя CompactFlash», стр. 17-18

Прежде чем использовать энергонезависимую память

Сохранение и загрузка имеют следующие параметры:

Параметр:	Сохранение	Загрузка:
Сколько времени занимает сохранение или загрузка?	Если контроллер *не* использует карту 1784-CF64 Industrial CompactFlash, сохранение может занять до трех минут. Если контроллер использует карту CompactFlash, сохранение происходит значительно быстрее (меньше минуты).	несколько секунд
В каком режиме(ах) контроллера можно сохранить или загрузить проект?	программный режим	
Можно ли перевести контроллер в режим онлайн во время сохранения или загрузки?	нельзя	
В каком состоянии находится ввод/вывод во время сохранения или загрузки?	ввод/вывод остается в состоянии, сконфигурированном для программного режима.	

Выбор контроллера, имеющего энергонезависимую память

Следующие контроллеры Logix5000 имеют энергонезависимую память для сохранения проектов.

Тип контроллера:	Номер по каталогу:	Ревизия микропрограммного обеспечения:	Требуется карта памяти 1784-CF64 Industrial CompactFlash:
CompactLogix5320	1769-L20	10.x или более поздняя	нет
CompactLogix5330	1769-L30	10.x или более поздняя	нет
CompactLogix5331	1769-L31	13.x или более поздняя	да
CompactLogix5332E	1769-L32E	13.x или более поздняя	да
CompactLogix5335CR	1769-L35CR	13.x или более поздняя	да
CompactLogix5335E	1769-L35E	12.x или более поздняя	да
CompactLogix5555	1756-L55M22	10.x или более поздняя	нет
	1755-L55M23	8.x или более поздняя	нет
	1756-L55M24	8.x или более поздняя	нет
CompactLogix5560M03SE	1756-L60M03SE	13.x или более поздняя	да
CompactLogix5561	1756-L61	12.x или более поздняя	да
CompactLogix5562	1756-L62	12.x или более поздняя	да
CompactLogix5563	1756-L63	11.x или более поздняя	да
DriveLogix5720	различные	10.x или более поздняя	нет
DriveLogix5730	различные	13.x или более поздняя	да
DriveLogix5433	1794-L33	10.x или более поздняя	нет
DriveLogix5434 Series B	1794-L34/B	11.x или более поздняя	нет

Предотвращение основной ошибки при загрузке

Если основная и неосновная ревизии проекта в энергонезависимой памяти не совпадают с основной и неосновной ревизиями контроллера, то во время загрузки *может* произойти основная ошибка.

Если контроллер:	То:
не использует карту CompactFlash	Убедитесь, что основная и неосновная ревизии проекта в энергонезависимой памяти совпадают с основной и неосновной ревизиями контроллера. Энергонезависимая память контроллера сохраняет только проект. Она не сохраняет микропрограммное обеспечение контроллера.
использует карту CompactFlash	Карта CompactFlash сохраняет микропрограммное обеспечение для проектов ? 12.0. В зависимости от текущей ревизии контроллера, вы сможете использовать карту CompactFlash для обновления микропрограммного обеспечения контроллера и загрузки проекта. См. «Определение того, каким образом производить обновление микропрограммного обеспечения» на стр. 17-6.

Форматирование карты CompactFlash

Когда вы сохраняете проект на карте памяти 1784-CF64 Industrial CompactFlash, то при необходимости контроллер форматирует эту карту.

<table>
<tr><th>Если ревизия вашего проекта:</th><th>То:</th></tr>
<tr><td>11.x</td><td>Карта CompactFlash использует специальный формат:
Используйте только контроллер Logix5000 для сохранения проекта на карте CompactFlash. Не используйте считыватель CompactFlash для чтения с карты или записи на нее с помощью компьютера.
Сохраняйте на карте CompactFlash лишь один проект Logox5000, без каких-либо других данных.
Когда вы сохраняете проект на карте CompactFlash, вы перезаписываете все содержимое этой карты. Другими словами, вы теряете все, что находится на карте в данный момент.</td></tr>
<tr><td>⩾ 12.0</td><td>Карта CompactFlash использует файловую систему FAT16.<table><tr><th>Если карта:</th><th>То контроллер:</th></tr><tr><td>уже отформатирована для файловой системы FAT16</td><td>оставляет существующие данные
создает папки и файлы для проекта и микропрограммного обеспечения</td></tr><tr><td>не отформатирована для файловой системы FAT16</td><td>удаляет существующие данные
форматирует карту для файловой системы FAT16
создает папки и файлы для проекта и микропрограммного обеспечения</td></tr></table>Когда карта CompactFlash отформатирована для файловой системы FAT16:
Карта CompactFlash сохраняет множество проектов и связанное с ними микропрограммное обеспечение.
Если карта CompactFlash уже содержит проект с таким именем, при сохранении проект на карте CompactFlash будет перезаписан.
Карта CompactFlash загружает те проекты, которые сохранялись последними.
Для ревизии ⩾ 12.0 вы также можете использовать считыватель CompactFlash для считывания и манипуляций с файлами на карте CompactFlash. Обратитесь к разделу «Использование считывателя CompactFlash» на стр. 17-18.</td></tr>
</table>

Определение того, каким образом производить обновление микропрограммного обеспечения

В следующей таблице указываются способы и меры предосторожности при обновлении микропрограммного обеспечения контроллера, имеющего энергонезависимую память.

<table>
<tr><th>Если:</th><th>То:</th></tr>
<tr><td>Вы имеете все ниже перечисленные условия:
☐ Контроллер использует карту 1784-CF64 Industrial CompactFlash.
☐ Проект на карте CompactFlash имеет ревизию ⩾ 12.0.
☐ Проект на карте CompactFlash имеет опцию Load Image (Загрузить образ) = On Power Up (При включении питания) или On Corrupt Memory (При повреждении памяти).
☐ Если используется контроллер 1756-L63, его микропрограммное обеспечение имеет одну из следующих ревизий:
☐ Для контроллера, который только что достали из коробки, ревизия ⩾ 1.4. (Ищите надпись F/W REV. на боковой стенке контроллера или его коробки).
☐ Для уже используемого контроллера ревизия ⩾ 12.0.</td><td>Обновите микропрограммное обеспечение посредством одного из следующего:
• карты CompactFlash
• программного обеспечения RSLogix 5000
• программного обеспечения ControlFlash
Чтобы обновить микропрограммное обеспечение и загрузить проект посредством карты CompactFlash:
1. Установите карту в контроллер:
2. Если опция Load Image (Загрузить образ) = On Corrupt Memory (При повреждении памяти) и контроллер содержит проект, отсоедините аккумуляторную батарею от контроллера.
3. Включите или выключите и включите питание контроллера.
Если вы используете программное обеспечение RSLogix 5000 или ControlFlash для обновления микропрограммного обеспечения:
1 Во время обновления контроллер устанавливает опцию Load Image (Загрузить образ) карты CompactFlash на значение User Initiated (По инициативе пользователя). Чтобы избежать этого, снимите карту с контроллера.
2. После обновления микропрограммного обеспечения сохраните проект снова в энергонезависимой памяти. Это обеспечит соответствие ревизии проекта в энергонезависимой памяти ревизии контроллера.</td></tr>
<tr><td>У вас не выполняются все вышеперечисленные условия:</td><td>Обновите микропрограммное обеспечение посредством одного из следующего:
• программное обеспечение RSLogix 5000
• программное обеспечение ControlFlash
Соблюдайте следующие меры предосторожности:
1 Перед обновлением микропрограммного обеспечения:
<table>
<tr><th>Если контроллер:</th><th>То:</th></tr>
<tr><td>не использует карту CompactFlash</td><td>Сохраните проект в автономном файле. Когда вы обновляете микропрограммное обеспечение контроллера, вы стираете содержимое энергонезависимой памяти (ревизия 10.x или более поздняя).</td></tr>
<tr><td>использует карту CompactFlash</td><td>Выполните одно из следующих действий:
• Снимите карту CompactFlash с контроллера.
• Проверьте опцию Load Image (Загрузить образ) карты CompactFlash. Если она установлена на On Power Up (При включении питания) или On Corrupt Memory (При повреждении памяти), то в первый раз сохраните проект с опцией Load Image, установленной на User Initiated (По инициативе пользователя).
В противном случае может возникнуть основная ошибка при обновлении микропрограммного обеспечения контроллера. Это связано с тем, что опции On Power Up и On Corrupt Memory заставляют контроллер загружать проект из энергонезависимой памяти. Несоответствие ревизий микропрограммного обеспечения после загрузки является причиной основной ошибки.</td></tr>
</table>
2. После обновления микропрограммного обеспечения вновь сохраните проект в энергонезависимой памяти. Это обеспечит соответствие ревизии проекта в энергонезависимой памяти ревизии контроллера.</td></tr>
</table>

Выбор времени загрузки образа

У вас есть несколько опций, позволяющих выбрать, когда (при каких условиях) загружать проект обратно в пользовательскую память (RAM) контроллера:

Если вы хотите загружать проект:	**То выберите:**	**Примечания:**

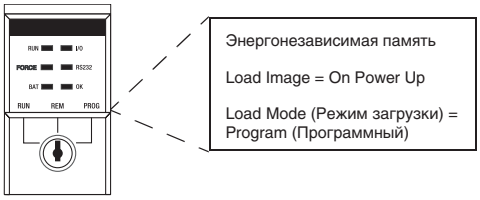
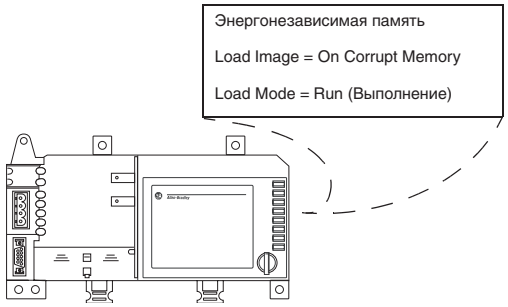
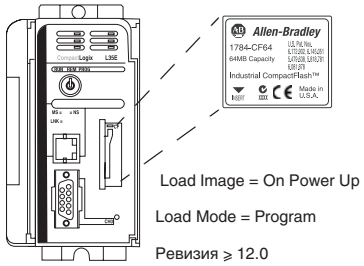
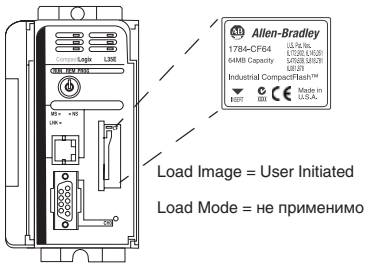
всякий раз, когда вы включаете или выключаете и включаете питание шасси	On Power Up (При включении питания)	• Во время выключения и последующего включения питания вы потеряете все внесенные в режиме онлайн изменения, значения тегов и сетевое расписание, которые вы не сохранили в энергонезависимой памяти.
		• Карта 1784-CF64 Industrial CompactFlash также может изменить микропрограммное обеспечение контроллера.
		  • Это происходит, если и ревизия проекта на карте CompactFlash, и ревизия микропрограммного обеспечения контроллера ⩾ 12.0.
		  • Для получения более подробной информации обратитесь к разделу «Определение того, каким образом производить обновление микропрограммного обеспечения» на стр. 17-6.
		• Вы всегда можете использовать программное обеспечение RSLogix 5000 для загрузки проекта.

всякий раз, когда в контроллере нет проекта и вы включаете или выключаете и включаете питание шасси	On Corrupt Memory (При повреждении памяти)	• Например, если аккумуляторная батарея разряжается и контроллер теряет питание, проект стирается из памяти. Когда подача питания восстанавливается, эта опция загрузки загружает проект обратно в контроллер.
		• Карта 1784-CF64 Industrial CompactFlash также может изменить микропрограммное обеспечение контроллера.
		  • Это происходит, если и ревизия проекта на карте CompactFlash, и ревизия микропрограммного обеспечения контроллера ⩾ 12.0.
		  • Для получения более подробной информации обратитесь к разделу «Определение того, каким образом производить обновление микропрограммного обеспечения» на стр. 17-6.
		• Вы всегда можете использовать программное обеспечение RSLogix 5000 для загрузки проекта.

только посредством программного обеспечения RSLogix 5000	User Initiated (По инициативе пользователя)	

Примеры

Здесь представлено несколько примеров использования различных вариантов загрузки:

Пример:	Описание:
1. 	1 Вы обновляете микропрограммное обеспечение контроллера до желаемой ревизии. 2. Вы сохраняете проект для контроллера в энергонезависимой памяти. 3. Когда вы включаете питание контроллера после инсталляции, данный проект загружается в контроллер. 4. Контроллер остается в программном режиме.
2. 	1 Вы сохраняете проект для контроллера в энергонезависимой памяти. (Основная и неосновная ревизии микропрограммного обеспечения в контроллере соответствуют основной и неосновной ревизиям проекта в энергонезависимой памяти). 2. Если аккумуляторная батарея контроллера полностью разрядилась и питание контроллера прекратилось, проект стирается из памяти контроллера. 3. Когда питание восстанавливается, проект автоматически загружается в контроллер, и контроллер возвращается в режим выполнения.
3. 	1 Контроллер выходит из строя. 2. Вы снимаете карту CompactFlash. 3. Вы заменяете вышедший из строя контроллер новым. 4. Вы заменяете карту CompactFlash. 5. Когда вы включите питание, и микропрограммное обеспечение, и проект загрузятся в контроллер. Контроллер остается в программном режиме.
4. 	1 Вы хотите загрузить другой проект в ваш контроллер. 2. Карта CompactFlash содержит требуемый проект. 3. Установив карту CompactFlash в контроллер, вы используете программное обеспечение RSLogix 5000 для загрузки проекта в контроллер.

Сохранение проекта

В этой задаче вы сохраняете проект в энергонезависимой памяти контроллера..

ВАЖНО Во время сохранения все активные оси сервосистемы выключаются. Перед сохранением проекта убедитесь, что это *не* приведет к непредвиденному перемещению оси.

Перед сохранением проекта:

- внесите все требуемые изменения в логику
- загрузите проект в контроллер
- составьте расписание для ваших сетей ControlNet

Чтобы сохранить проект:

☐ Сконфигурируйте операцию сохранения (Store Operation)

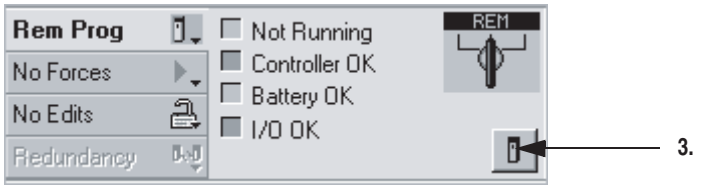
☐ Сохраните проект (Project)

☐ Сохраните онлайновый проект (Online Project)

Конфигурирование операции сохранения (Store Operation)

1.Переведите контроллер в режим онлайн.

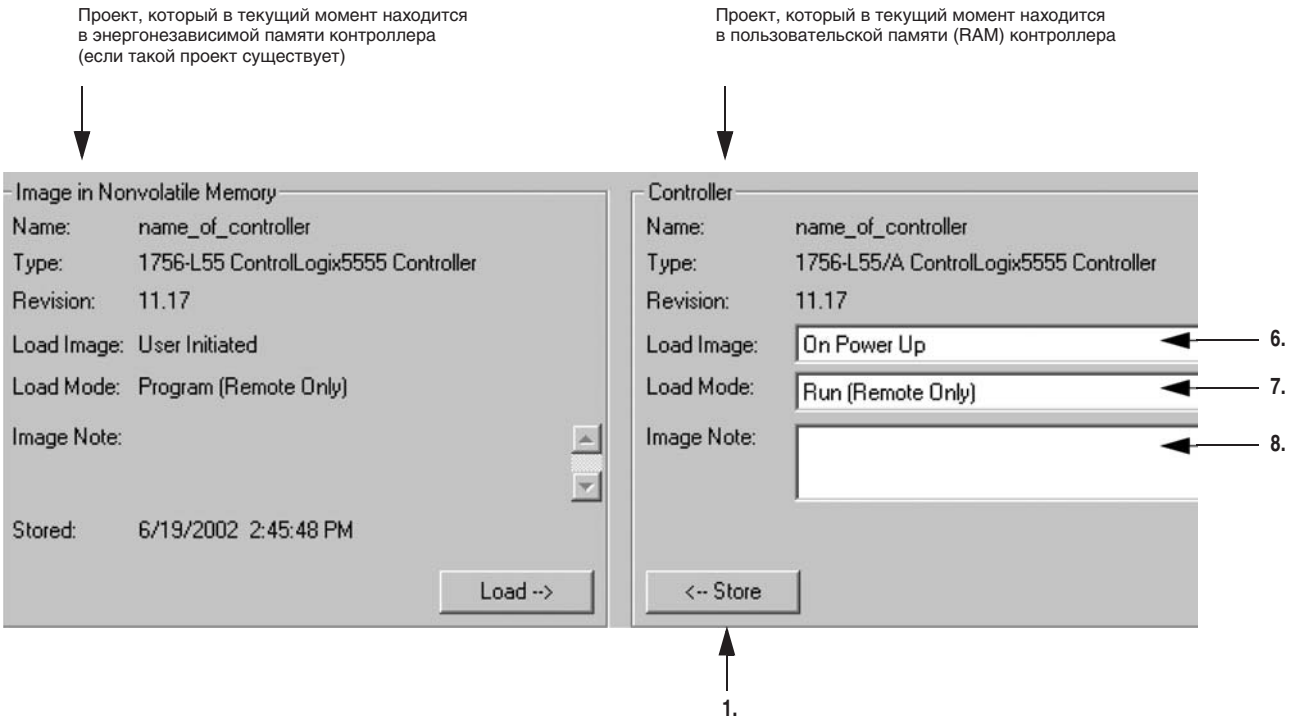
2. Установите для контроллера программный режим (Rem Program или Program).



3. На панели инструментов режима онлайн (Online) нажмите на кнопку свойств контроллера.

4. Нажмите на закладку *Nonvolatile Memory* (Энергонезависимая память).

5. Выберите *Load/Store* (Загрузить/сохранить).



6. Выберите, когда (при каких условиях) загружать проект обратно в пользовательскую память (RAM) контроллера.

7. Какую опцию загрузки образа вы использовали в шаге 6?

Если:	**То:**
On Power Up (При включении питания)	Выберите режим, в который будет переходить контроллер после загрузки: • удаленный программный • удаленный выполнения Чтобы контроллер перешел в этот режим после загрузки, установите кнопочный переключатель контроллера в положение REM.
On Corrupt Memory (При повреждении памяти)	
User Initiated (По инициативе пользователя)	Перейдите к шагу 8.

8. По желанию введите примечание, описывающее сохраняемый проект.

Сохранение проекта

1. Выберите <- *Store* (Сохранить).

   Появится диалоговое окно с запросом подтверждения сохранения.

2. Чтобы сохранить проект, выберите *Yes*.

   Во время сохранения произойдут следующие события:

   - На передней панели контроллера светодиодный индикатор подтверждения загорится в следующей последовательности:

     мигающий зеленый => непрерывный красный => непрерывный зеленый

   - Программное обеспечение RSLogix 5000 перейдет в режим оффлайн.
   - Появится диалоговое окно, сообщающее вам, что сохранение выполняется.

3. Выберите *OK*.

   Когда сохранение завершится, вы останетесь в режиме оффлайн.

Сохранение проекта онлайн

1. Переведите контроллер в режим онлайн.
2. Сохраните проект.

Загрузка проекта

В этой задаче вы используете программное обеспечение RSLogix 5000 для загрузки проекта из энергонезависимой памяти.

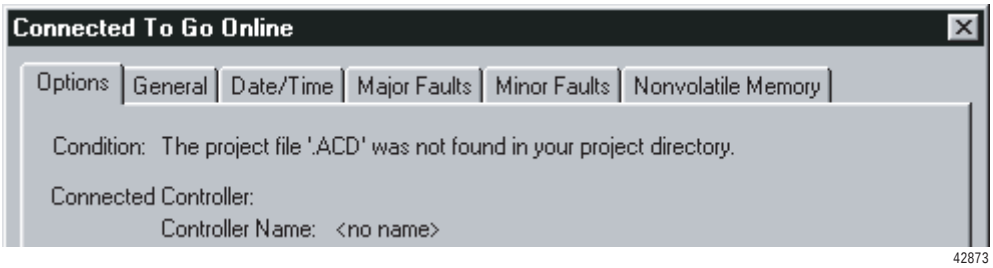
.

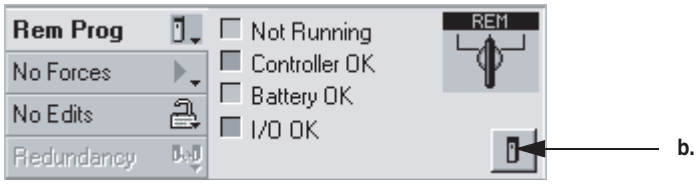
ВНИМАНИЕ

Во время загрузки все активные оси сервосистемы выключаются. Перед загрузкой проекта убедитесь, что это *не* приведет к непредвиденному перемещению оси.

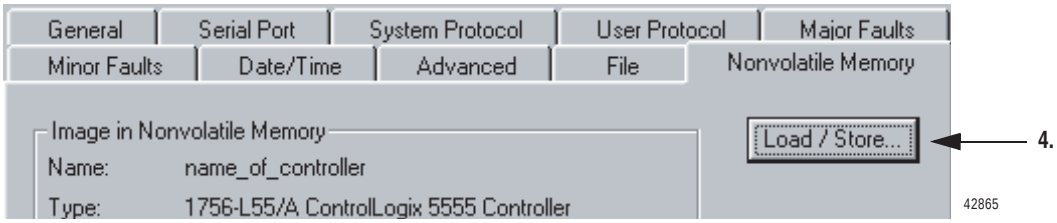
Шаги:

1. Переведите контроллер в режим онлайн.
2. Открылось ли следующее диалоговое окно?

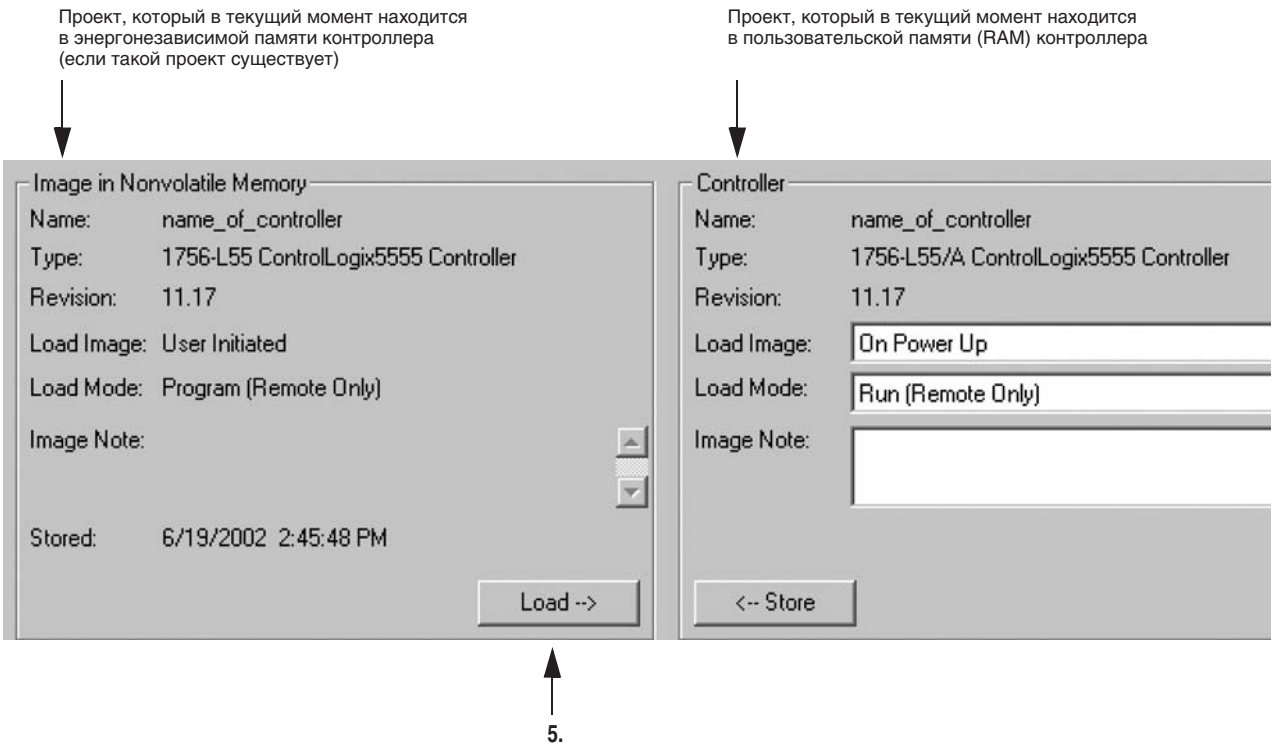


Если:	То:
Нет	c. Переведите контроллер в программный режим (Rem Program или Program).  d. На панели инструментов режима онлайн нажмите на кнопку свойств контроллера
Да	Переведите контроллер в программный режим (Rem Program или Program). Используйте одно из следующего: • закладку *General* (Общее) в диалоговом окне *Connected To Go Online* (Подключен для перехода в режим онлайн) • кнопочный переключатель на передней панели контроллера

3. Нажмите на закладку *Nonvolatile Memory* (Энергонезависимая память).



4. Выберите *Load/Store* (Загрузить/сохранить).



5. Выберите *Load –>* (Загрузить).

Появится диалоговое окно с запросом подтверждения загрузки.

6. Чтобы загрузить проект из энергонезависимой памяти, выберите *Yes*.

Во время загрузки произойдут следующие события:

- На передней панели контроллера светодиодный индикатор подтверждения загорится в следующей последовательности:

Если загрузка:	То индикатор будет гореть:
не включает микропрограммное обеспечение	непрерывным красным => непрерывным зеленым
включает микропрограммное обеспечение	мигающим зеленым => непрерывным красным => непрерывным зеленым

- Программное обеспечение RSLogix 5000 перейдет в режим оффлайн.

Когда загрузка завершится, вы останетесь в режиме оффлайн. Если вы хотите перейти в режим онлайн, вам придется сделать это вручную.

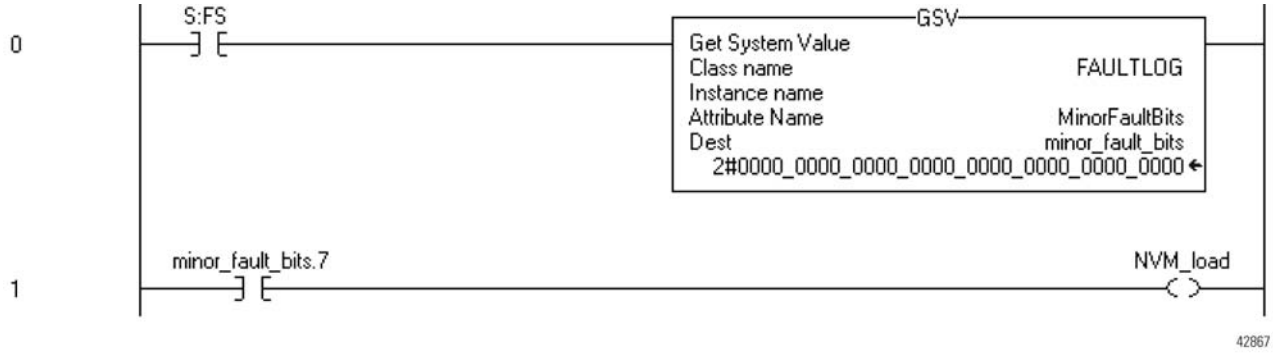
Пометка для загрузки

Когда контроллер загружает проект из энергонезависимой памяти, он передает следующую информацию:

- регистрирует неосновную ошибку (тип 7, код 49)
- устанавливает объект FAULTLOG (журнал регистрации ошибок), атрибут MinorFaultBits (биты неосновных ошибок), бит 7

Если вы хотите, чтобы ваш проект имел пометку о том, что он загружался из энергонезависимой памяти, используйте следующую релейную логику:

При первом сканировании проекта (*S:FS* включен) инструкция GSV получает объект FAULTLOG, атрибут MinorFaultBit и сохраняет это значение в *minor_fault_bits*. Если бит 7 установлен, контроллер загрузил проект из энергонезависимой памяти.



Где:	Является:
`minor_fault_bits`	тегом, в котором хранится объект FAULTLOG, атрибут MinorFaultBits. Тип данных – DINT.
`NVM_laod`	тегом, который указывает на то, что контроллер загрузил проект из энергонезависимой памяти.

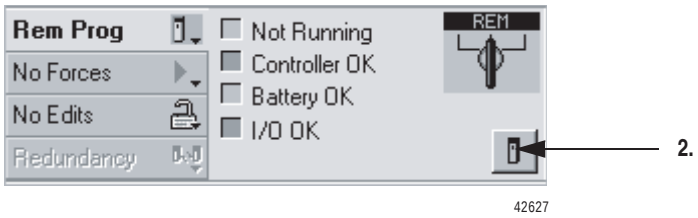
Очистка энергонезависимой памяти

Чтобы удалить проект из энергонезависимой памяти, выполните следующие действия:

- ☐ Проверьте текущую опцию Load Image (Загрузить образ)
- ☐ Измените опцию Load Image
- ☐ Удалите проект из контроллера
- ☐ Сохраните пустой образ

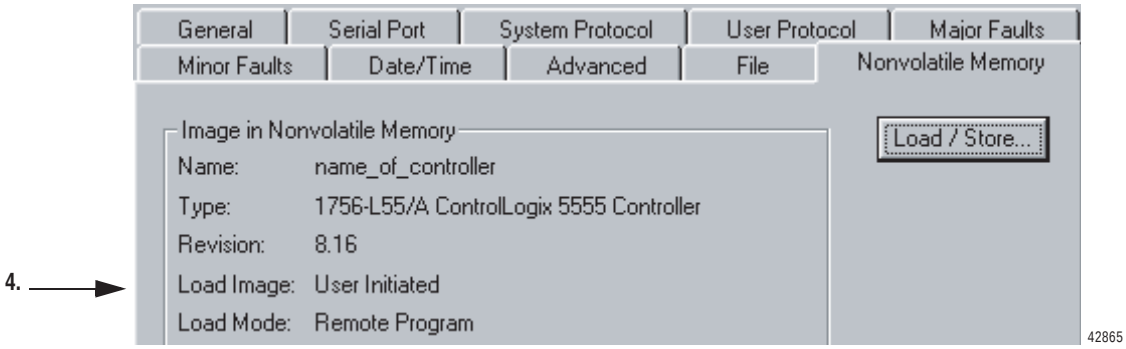
Проверка текущей опции Load Image

1. Переведите контроллер в режим онлайн.



2. На панели инструментов режима онлайн нажмите на кнопку свойств контроллера.

3. Щелкните по закладке *Nonvolatile Memory* (Энергонезависимая память).



4. Опция *Load Image* установлена на *User Initiated (По инициативе пользователя)*?

Если:	То:
Нет	Перейдите к разделу «Изменение опции Load Image» на стр. 17-16.
Да	Перейдите к разделу «Удаление проекта из контроллера» на стр. 17-16.

Изменение опции Load Image

1. Выберите *Load/Store* (Загрузить/сохранить).
2. В выпадающем списке *Load Image* выберите *User Initiated* (По инициативе пользователя).
3. Выберите <- *Store* (Сохранить).

   Появится диалоговое окно с запросом подтверждения сохранения.
4. Чтобы сохранить проект, выберите *Yes*.

   Появится диалоговое окно, сообщающее о том, что сохранение выполняется.
5. Нажмите *OK*.
6. Дождитесь, чтобы светодиодный индикатор подтверждения на передней панели контроллера непрерывно горел зеленым. Это означает, что сохранение завершено.

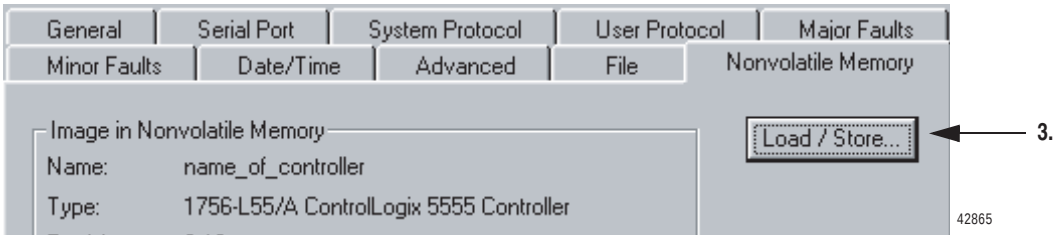
Удаление проекта с контроллера

1. Отсоедините аккумуляторную батарею от контроллера.
2. Включите и выключите питание шасси.
3. Вновь подсоедините аккумуляторную батарею к контроллеру.

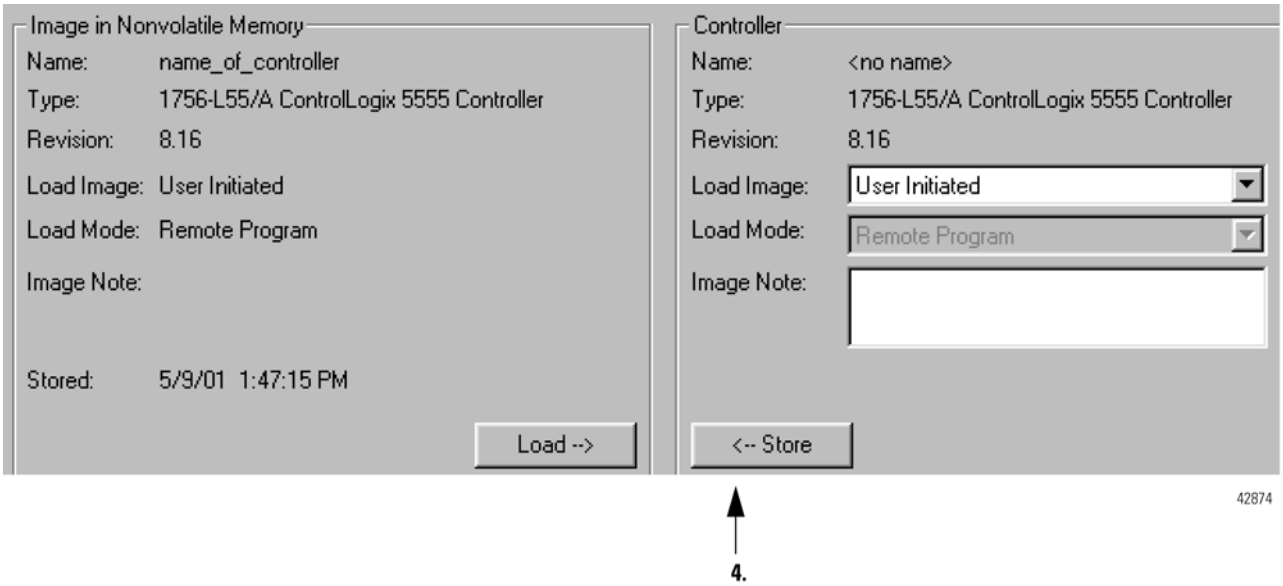
Сохранение пустого образа

1. Переведите контроллер в режим онлайн.

   Откроется диалоговое окно Connected To Go Online (Подключен для перехода в режим онлайн).
2. Нажмите на закладку *Nonvolatile Memory* (Энергонезависимая память).



3. Выберите *Load/Store* (Загрузить/сохранить).



4. Выберите <- *Store* (Сохранить).

Появится диалоговое окно с запросом подтверждения сохранения.

5. Чтобы сохранить проект, выберите *Yes*.

Во время сохранения произойдут следующие события:

- На передней панели контроллера светодиодный индикатор подтверждения загорится в следующей последовательности:

  мигающий зеленый => красный => зеленый

- Программное обеспечение RSLogix 5000 перейдет в режим оффлайн.
- Появится диалоговое окно, сообщающее вам, что сохранение выполняется.

6. Нажмите *OK*.

Когда сохранение завершится, вы останетесь в режиме оффлайн. Если вы хотите перейти в режим онлайн, вам придется сделать это вручную.

Использование считывателя CompactFlash

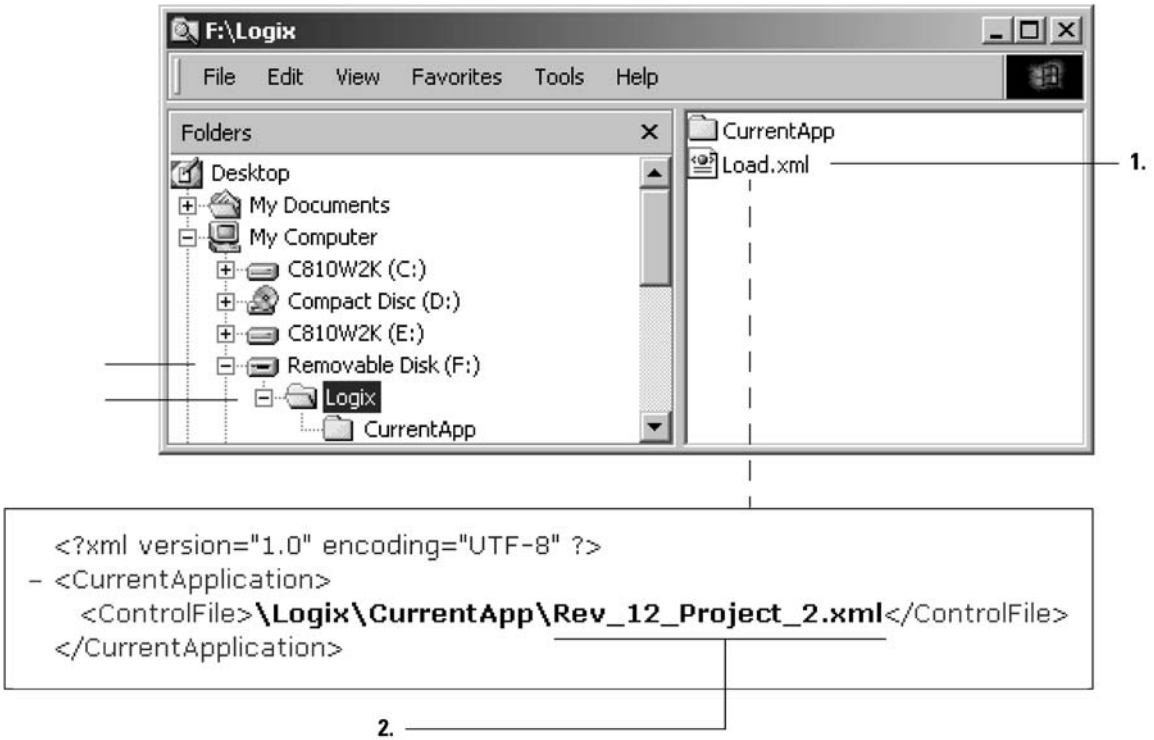
Если ревизия вашего проекта или проектов на карте CompactFlash ? 12.0, то карта отформатирована с помощью файловой системы FAT16.

- Обычно вам *не* требуется управлять файлами на карте CompactFlash. Карта автоматически загружает проект, который вы сохранили последним.
- В целях обеспечения дополнительной гибкости, файловая система также позволяет вам:
  - ☐ Вручную изменять информацию о том, какой проект будет загружаться с карты CompactFlash
  - ☐ Вручную изменять параметры загрузки для проекта

Изменение вручную информации о том, какой проект будет загружаться с карты CompactFlash

На карте CompactFlash сохраняется множество проектов. По умолчанию контроллер загружает проект, который вы сохранили последним, в соответствии с опциями загрузки этого проекта.

Чтобы задать другой проект для загрузки с карты CompactFlash, вам необходимо отредактировать файл *Load.xml* на карте.



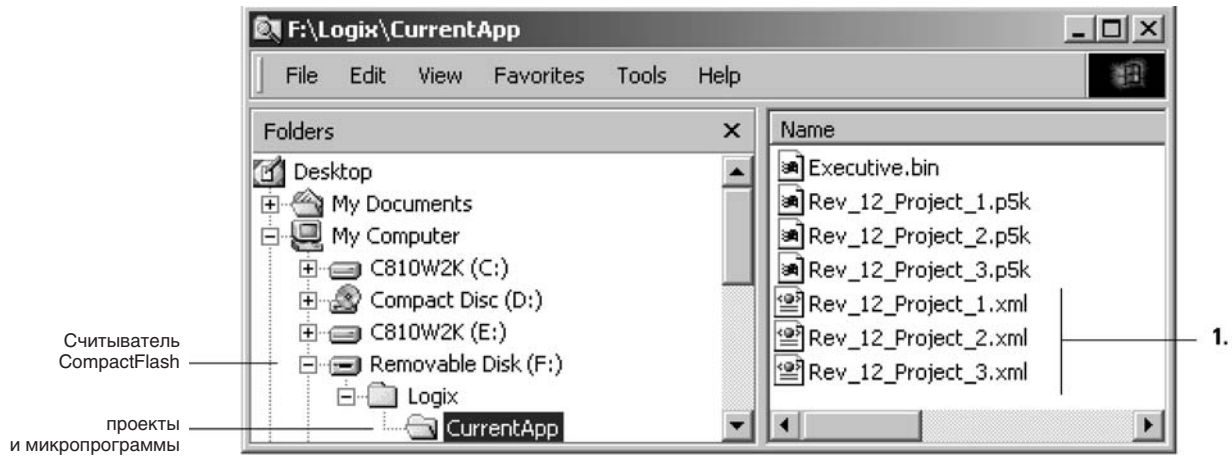
1. Чтобы изменить информацию о том, какой проект будет загружаться с карты, откройте *Load.xml*. Используйте текстовый редактор для открытия этого файла.
2. Отредактируйте имя проекта, который вы хотите загрузить.
   - Используйте имя файла XML, находящегося в папке *CurrentApp*.
   - В папке *CurrentApp* проект состоит из файла XML и файла P5K.

Изменение вручную параметров загрузки для проекта

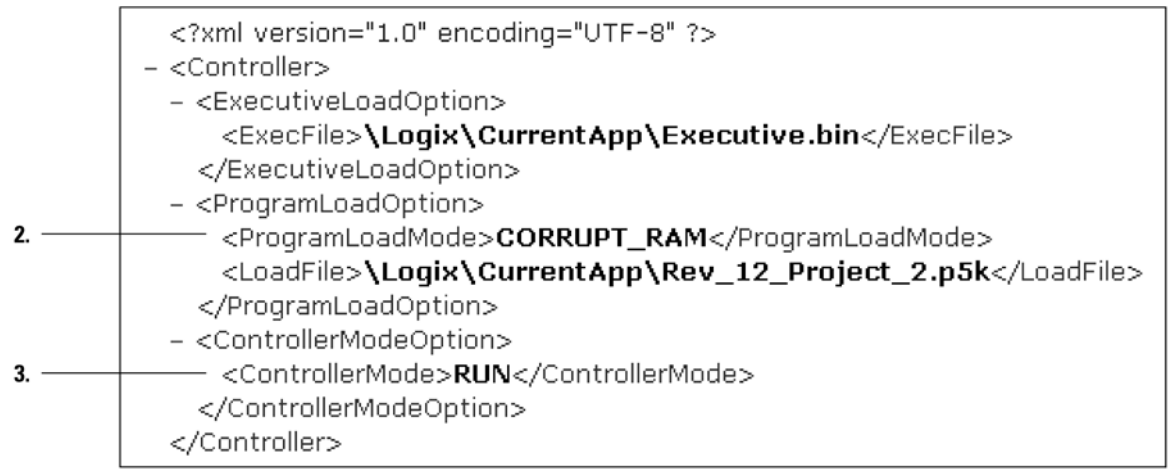
Когда вы сохраняете проект в энергонезависимой памяти, вы задаете:

- когда проект должен загружаться (*On Power Up* (При включении питания), *On Corrupt Memory* (При повреждении памяти), *User Initiated* (По инициативе пользователя))
- режим, в который нужно установить контроллер (если кнопочный переключатель находится в положении REM и режим загрузки не *User Initiated*)

Чтобы задать другой проект для загрузки с карты CompactFlash, вам необходимо отредактировать файл *Load.xml* на карте.



1. Чтобы изменить параметры загрузки для проекта, откройте файл XML с таким же именем, как у проекта. Используйте текстовый редактор для открытия этого файла.



2. Отредактируйте опцию Load Image (Загрузка образа) проекта.

Если вы хотите установить опцию Load Image на:	**То введите:**
On Power Up	ALWAYS
On Corrupt Memory	CORRUPT_RAM
User Initiated	USER_INITIATED

3. Отредактируйте опцию Load Mode (Режим загрузки) проекта (не применяется, если опция Load Image установлена на *User Initiated*).

Если вы хотите установить опцию Load Mode на:	**То введите:**
Program (Remote Only) (Программный (только удаленный))	PROGRAM
Run (Remote Only) (Выполнения (только удаленный))	RUN

Обеспечение защиты проекта

Когда использовать эту процедуру

Используйте эту процедуру для управления доступом к вашему проекту. Для обеспечения защиты вашего проекта имеются следующие опции:

Если вы хотите:	То:	См. стр.:
запретить другим видеть логику внутри одной или нескольких процедур проекта	Используйте защиту исходного текста процедуры	18-1
присвоить различные уровни доступа к проекту, например, позволить: • инженерам иметь полный доступ • специалистам по техническому обслуживанию вносить ограниченные изменения • операторам только просматривать логику и данные	Используйте сервер безопасности RSI Security Server для защиты проекта	18-13

Вы можете использовать обе опции одновременно.

Использование защиты исходного текста процедуры

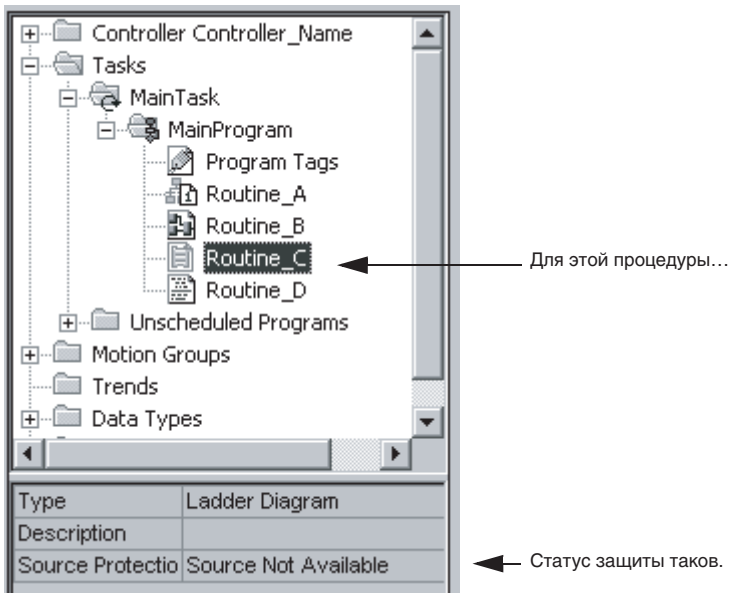
Чтобы ограничить доступ к процедуре, используйте программное обеспечение RSLogix 5000 для присвоения **ключа исходного текста** процедуре (обеспечения защиты процедуры).

- Чтобы обеспечить защиту процедуры, вам необходимо, прежде всего, активировать эту функцию программного обеспечения RSLogix 5000.
- Когда процедура защищена, компьютер будет требовать ключ исходного текста для редактирования, копирования или экспорта процедуры.
- Вы можете также задать, чтобы процедура была видимой или невидимой без ключа исходного текста.
- Независимо от наличия ключа исходного текста, вы всегда можете скачивать проект и выполнять все процедуры.
- Вы можете вновь получить доступ к защищенной процедуре с определенного компьютера при помощи одного из следующих методов:

  - Добавьте файл ключа исходного текста и укажите программному обеспечению RSLogix 5000 местоположение файла.

  - Создайте файл ключа исходного текста и вручную введите имя ключа.

Организатор контроллера показывает статус защиты процедуры:



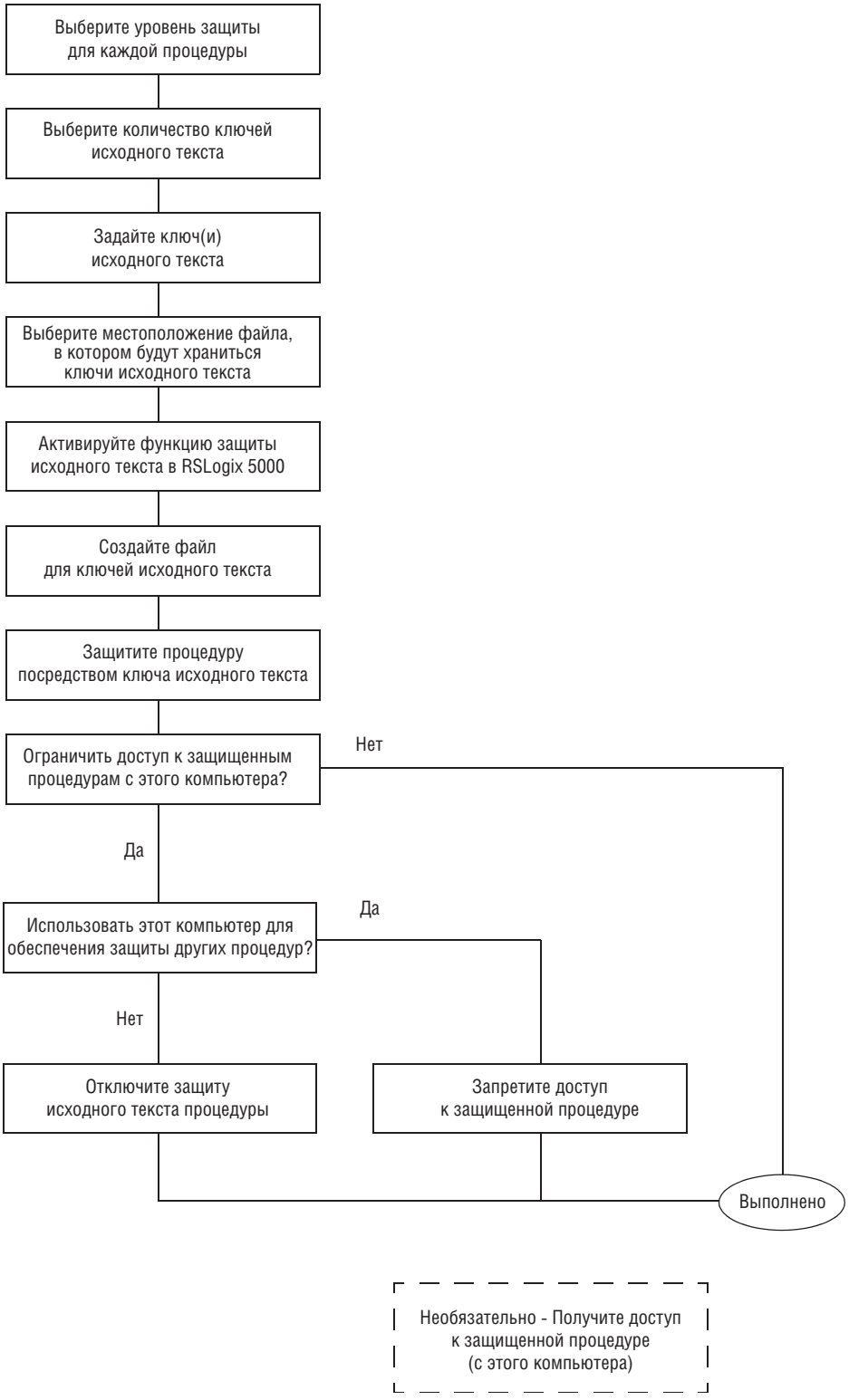
Если организатор контроллера показывает:	**То:**
Source Not Available (Исходный текст недоступен)	• Данной процедуре присвоен ключ исходного текста. • Чтобы открыть процедуру, вашему компьютеру потребуется ключ исходного текста для данной процедуры.
Source Not Available (Viewable) (Исходный текст недоступен (Может просматриваться)	• Данной процедуре присвоен ключ исходного текста. • Вы можете только открывать и просматривать эту процедуру. • Вы не можете ни вносить изменения, ни копировать какую-либо часть процедуры.
Source Available (Исходный текст доступен)	• Данной процедуре присвоен ключ исходного текста. • Вы имеете полный доступ к этой процедуре.
Source Available (Viewable) (Исходный текст доступен (Может просматриваться))	• Данной процедуре присвоен ключ исходного текста. • Вы имеете полный доступ к этой процедуре. • Те, кто не имеют ключа исходного текста, все равно могут просматривать эту процедуру.
ничего из выше перечисленного	• Данной процедуре не присвоен ключ исходного текста. • Вы имеете полный доступ к этой процедуре.

ВАЖНО

Если исходный текст процедуры недоступен, *нельзя* экспортировать проект.

- Файл экспорта (.L5K) содержит только те процедуры, исходный код которых доступен.
- Если вы экспортируете проект, в котором исходный код доступен *не* для всех процедур, вы *не* сможете восстановить весь проект.

Для присвоения ключей исходного текста и управления ими выполните следующие действия:



Выбор уровня защиты для каждой процедуры

Защита исходного текста обеспечивает защиту вашего проекта на уровне процедур. Вы можете защитить некоторые процедуры проекта, оставив остальные процедуры незащищенными (доступными всем). Вы можете также защитить процедуру, но при этом позволить всем ее просматривать.

Таблица 18.1 Опции защиты процедуры

Если вы хотите:	А также:	То:	
		Защитить процедуру?	**Разрешить просмотр?**
запретить следующее: • редактировать процедуру • изменять свойства процедуры • экспортировать процедуру	запретить следующее: • открывать (показывать) процедуру • производить поиск процедуры • пользоваться перекрестными ссылками внутри процедуры • распечатывать процедуру	да	нет
	никаких других ограничений	да	да
разрешить полный доступ к процедуре для всех	→	нет	

Выбор количества ключей исходного текста

Чтобы обеспечить защиту процедуры, присвойте этой процедуре **ключ исходного текста**. Вы можете повторно использовать ключ исходного текста сколько угодно раз, как показано ниже.

Это:	**Дает вам:**
один ключ исходного текста для всех проектов	наименьшее количество ключей исходного текста (легче управлять, но защита хуже)
уникальный ключ исходного текста для каждого проекта	↓
уникальный ключ исходного текста для каждой процедуры в каждом проекте	наибольшее количество ключей источника (сложнее управлять, но защита лучше)

Выберите количество ключей исходного текста, которое бы оптимально сбалансировало вашу необходимость в защите с уровнем управления ключами исходного текста, который вы хотели бы на себя взять.

Задание ключа(ей) исходного текста

Ключи исходного текста подчиняются тем же правилам присвоения имен, что и другие компоненты RSLogix 5000, такие как процедуры, теги и модули. Следуйте этим правилам при присвоении имени ключу исходного текста:

- имя должно начинаться с буквы (A-Z или a-z) или знака подчеркивания (_)
- может содержать только буквы, цифры и знаки подчеркивания
- может включать не более 40 символов
- не должно включать несколько знаков подчеркивания (_) подряд или замыкающий знак подчеркивания (_)
- *не* чувствительно к регистру

Выбор местоположения файла, в котором будут храниться ключи исходного текста

Ключи исходного текста хранятся в файле ключей исходного текста (sk.dat). Файл ключей исходного текста отличается от файлов проекта RSLogix 5000 (.acd). Вы можете сохранить файл ключей исходного текста в любой папке по выбору.

Активация функции защиты исходного текста в RSLogix 5000

Чтобы использовать функцию защиты исходного текста процедуры, имеющуюся в программном обеспечении RSLogix 5000, вам необходимо ввести следующие регистрационные данные, которые активируют эту функцию:

Ключ:	Ввод значения:		
	Имя:	Тип:	Данные:
HKEY_CURRENT_USER\Software\Rockwell Software\RSLogix 5000\ProtectedRoutine	PTCRoutine	DWORD	1

Чтобы ввести регистрационные данные:

1. Вставьте ваш компакт-диск с программным обеспечением RSLogix 5000 в дисковод.
2. Запустите с диска следующий файл:

   language\Tools\Source Protection Tool\Enable Protected Routine Config.reg

   где:

   language – это язык вашего программного обеспечения. Например, для программного обеспечения на английском языке откройте папку ENU.

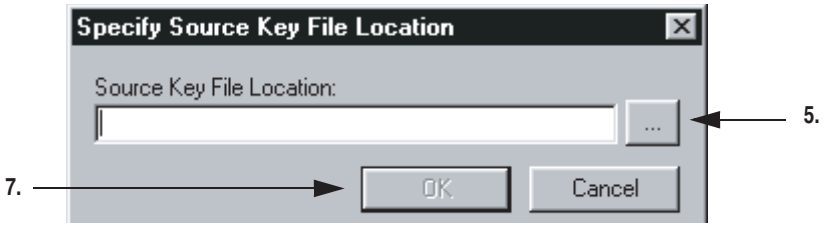
   Файл Enable Protected Routine Config.reg введет требуемые регистрационные данные.

Создание файла для ключей исходного текста

1. Откройте проект RSLogix 5000, для которого вы хотите создать защиту.
2. Из меню *Tools* (Сервис), выберите *Security* (Защита) => *Configure Source Protection* (Сконфигурировать защиту исходного текста).
3. Программное обеспечение RSLogix 5000 просит вас задать местоположение для файла ключей исходного текста?

Если:	То:
Нет	Ваш компьютер уже имеет файл ключей исходного текста. Перейдите к разделу «Защита процедуры при помощи ключей исходного текста» на стр. 18-7.
Да	Перейдите к шагу 4.

4. Выберите *Yes*.



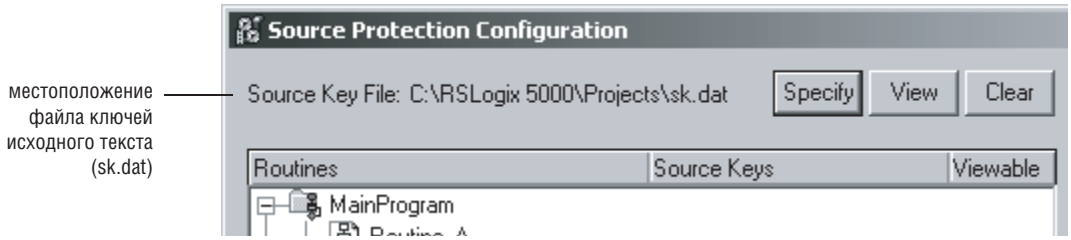
5. Нажмите на ...

6. Выберите папку, в которой будет сохранен файл, и нажмите *OK*.

7. Нажмите *OK*.

В диалоговом окне появится вопрос, хотите ли вы создать файл ключей исходного текста (sk.dat).

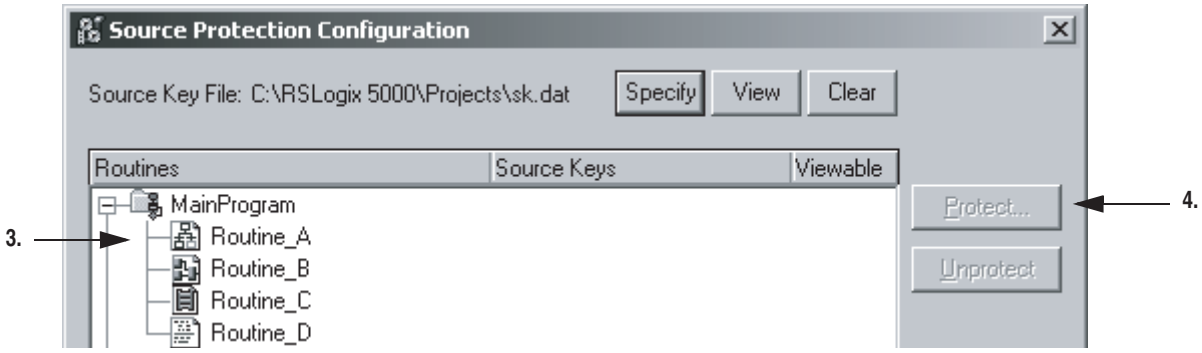
8. Выберите *Yes*.



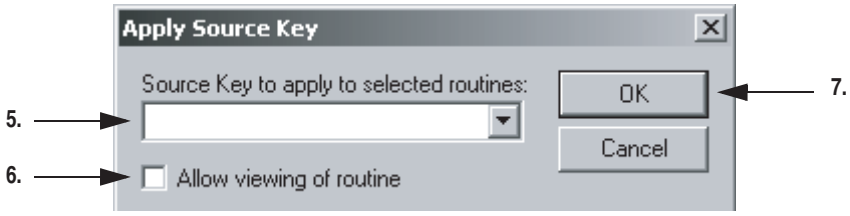
Защита процедуры при помощи ключей исходного текста

1. Откройте проект RSLogix 5000, для которого вы хотите создать защиту.

2. Из меню *Tools* (Сервис) выберите *Security* (Защита) => *Configure Source Protection* (Сконфигурировать защиту исходного текста).



3. Выберите процедуру или процедуры, для которых вы хотите создать защиту.
4. Нажмите *Protect* (Защитить).



5. Введите **имя**, которое вы хотите использовать в качестве ключа исходного текста. Или выберите существующий ключ исходного текста из выпадающего списка.
6. Вы хотели бы, чтобы те, кто не имеет ключа исходного текста, могли открывать и просматривать процедуру?

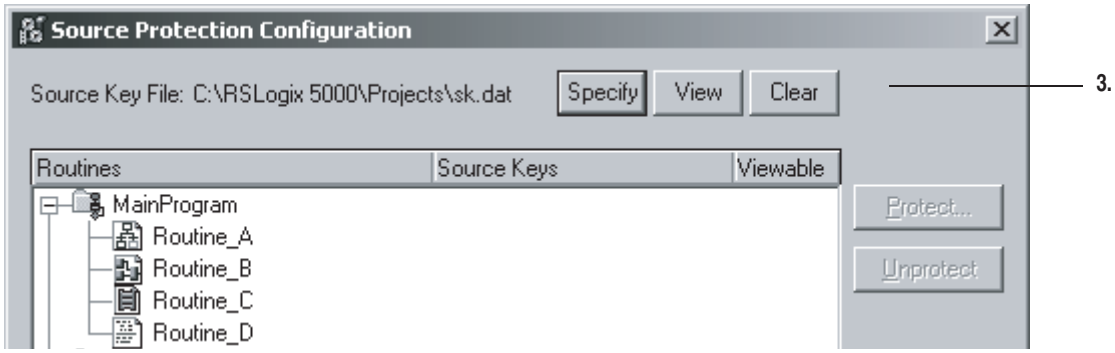
Если:	То:
Нет	Снимите флажок *Allow viewing of routine* (Разрешить просмотр процедуры) (установлен по умолчанию)
Да	Установите флажок *Allow viewing of routine*.

7. Нажмите *OK*.
8. Когда вы присвоите проекту необходимые ключи исходного текста, нажмите *Close* (Закрыть).
9. Из меню *File* (Файл) выберите *Save* (Сохранить).

Запрет доступа к защищенной процедуре

ВАЖНО Прежде чем вы удалите файл ключей исходного текста (sk.dat) из компьютера, запишите ключи исходного текста либо сделайте копию этого файла и храните его в надежном месте.

1. Откройте защищенный проект RSLogix 5000.
2. Из меню *Tools* (Сервис) выберите *Security* (Защита) => *Configure Source Protection* (Сконфигурировать защиту исходного текста).



3. Нажмите *Clear* (Очистить).

В диалоговом окне появится вопрос, хотите ли вы удалить файл ключей исходного текста (sk.dat).

4. Вы хотите удалить файл ключей исходного текста из компьютера (запретить дальнейший доступ к файлу)?

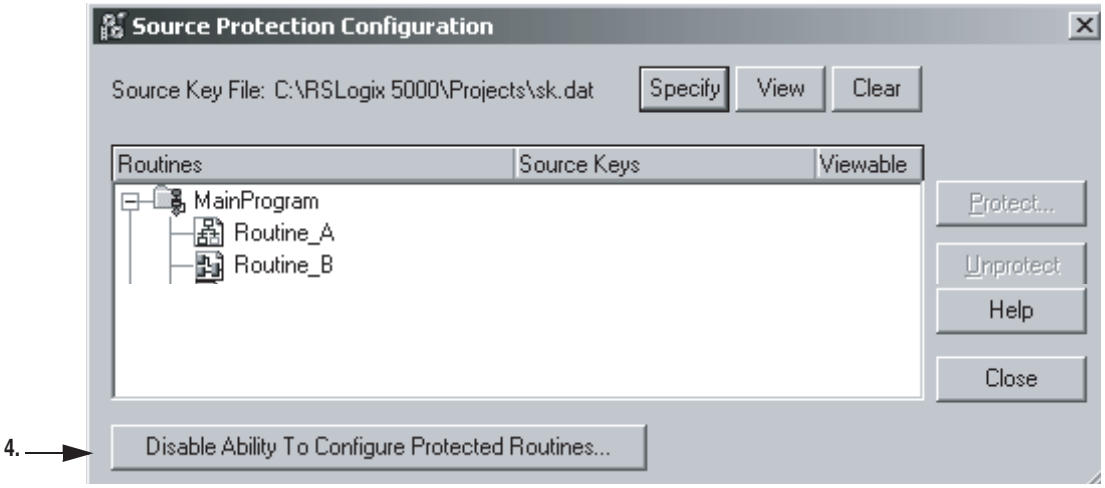
Если:	То:
Да	Выберите *Yes*.
Нет	Выберите *No*.

Отключение защиты исходного текста процедуры

ВАЖНО Прежде чем вы удалите файл ключей исходного текста (sk.dat) из компьютера, запишите ключи исходного текста либо сделайте копию этого файла и храните его в надежном месте.

1. Откройте защищенный проект RSLogix 5000.

2. Из меню *Tools* (Сервис) выберите *Security* (Защита) => *Configure Source Protection* (Сконфигурировать защиту исходного текста).



3. Нажмите *Disable Ability To Configure Protected Routines* (Заблокировать возможность конфигурировать защищенные процедуры).

   Появится диалоговое окно с запросом подтверждения действия.

4. Выберите *Yes*.

   В диалоговом окне появится вопрос, хотите ли вы удалить файл ключей исходного текста (sk.dat).

5. Вы хотите удалить файл ключей исходного текста из компьютера (запретить дальнейший доступ к файлу)?

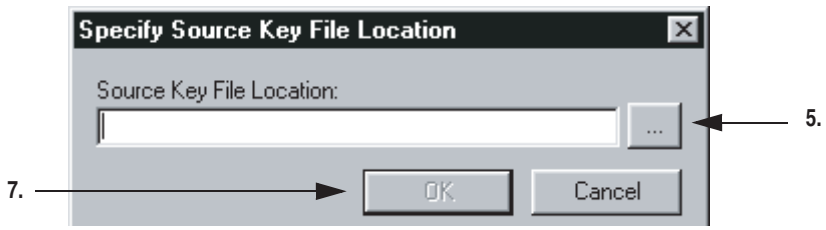
Если:	То:
Да	Выберите *Yes*.
Нет	Выберите *No*.

Получение доступа к защищенной процедуре

1. Откройте проект RSLogix 5000, содержащий защищенные процедуры.
2. Из меню *Tools* (Сервис) выберите *Security* (Защита) => *Configure Source Protection* (сконфигурировать защиту исходного текста).
3. Программное обеспечение RSLogix 5000 просит вас задать местоположение для файла ключей исходного текста?

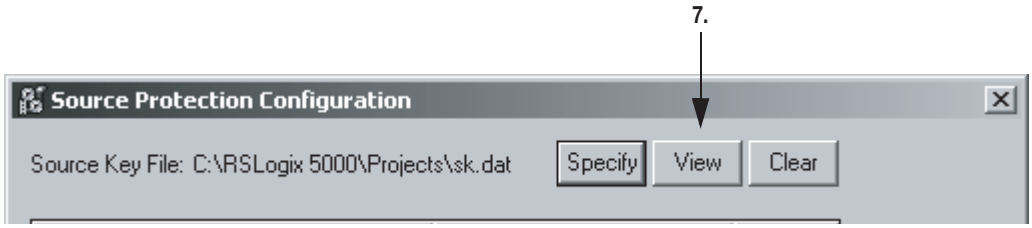
Если:	То:
Нет	Перейдите к шагу 7.
Да	Перейдите к шагу 4.

4. Выберите *Yes*.



5. Нажмите на ...
6. На этом компьютере уже есть файл ключей исходного текста (sk.dat)?

Если:	То:
Да	А: Выберите папку, которая содержит этот файл, и нажмите *OK*. Б: Нажмите *OK*.
Нет	А: Выберите папку, в которой вы хотите сохранить новый файл, и нажмите *OK*. В диалоговом окне появится вопрос, хотите ли вы создать файл ключей исходного текста (sk.dat). Б: Выберите *Yes*.



7. Нажмите *View* (Просмотр).

- Если вы получите приглашение выбрать программу, с помощью которой должен открываться файл, выберите программу обработки текстов, например, Notepad (Блокнот).
- Откроется файл sk.dat.

8. Введите имя ключа исходного текста. Чтобы ввести несколько ключей, вводите каждый ключ с новой строки.

sk.dat - Notepad
key1
key2
key3

9. Сохраните и закройте файл sk.dat.

Использование сервера безопасности RSI Security Server для защиты проекта

Программное обеспечение RSI Security Server позволяет вам управлять доступом к проектам RSLogix 5000. С помощью этого программного обеспечения вы можете настраивать доступ к проектам на основе:

- пользователя, который в данный момент зарегистрирован на рабочей станции
- проекта RSLogix 5000, к которому пользователь имеет доступ
- рабочей станции, с которой пользователь имеет доступ к проекту RSLogix 5000

Прежде чем использовать программное обеспечение Security Server для проектов RSLogix 5000, настройте это программное обеспечение:

- Установите программное обеспечение RSI Security Server
- Настройте DCOM
- Подключите Security Server для программного обеспечения RSLogix 5000
- Импортируйте файл RSLogix5000Security.bak
- Задайте глобальные действия для ваших пользователей
- Задайте действия проекта для ваших пользователей
- Добавьте пользователей
- Добавьте группы пользователей
- Назначьте глобальный доступ к программному обеспечению RSLogix 5000
- Назначьте действия проекта для новых проектов RSLogix 5000

После настройки программного обеспечения Security Server для проектов RSLogix 5000, выполните следующие действия для обеспечения защиты проекта:

- Создайте защиту проекта RSLogix 5000
- Назначьте доступ к проекту RSLogix 5000
- Обновите программное обеспечение RSLogix 5000, если необходимо

Установка программного обеспечения RSI Security Server

ВАЖНО

Если на момент установки программного обеспечения Security Server на вашем компьютере уже установлено программное обеспечение RSLogix 5000, подключите защиту для программного обеспечения RSLogix 5000, когда вы получите соответствующий запрос.

Обратитесь к руководству *«Достижение результатов с помощью программного продукта Rockwell Security Server» (автономное издание) (Getting Results with Rockwell Software's Security Server)*, которое поставляется в комплекте с программным обеспечением RSI Security Server.

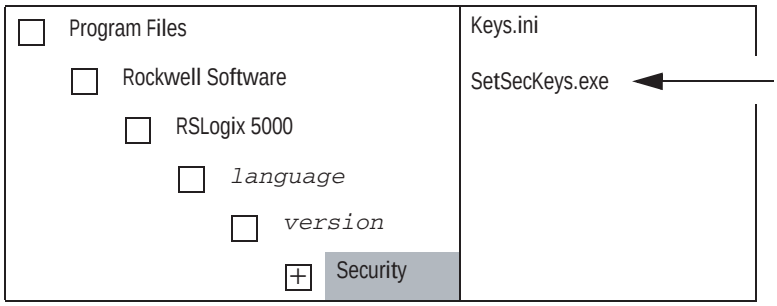
Настройка DCOM

Обратитесь к руководству *«Достижение результатов с помощью программного продукта Rockwell Security Server» (автономное издание) (Getting Results with Rockwell Software's Security Server)*, поставляемому в комплекте с программным обеспечением RSI Security Server.

Подключение Security Server для программного обеспечения RSLogix 5000

Вы установили Security Server *до* установки программного обеспечения RSLogix 5000?

Если:	То:
Да	Перейдите к шагу 1.
Нет	Перейдите к разделу «Импорт файла RSLogix5000Security.bak» на стр. 18-15.

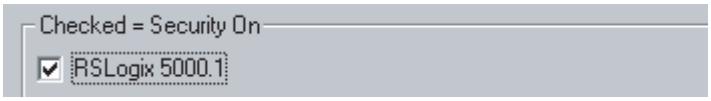


Где:	Является:
`language`	языком вашего программного обеспечения. Например, для программного обеспечения на английском языке откройте папку ENU.
`version`	версией вашего программного обеспечения, например v10

Откроется диалоговое окно Locate Project File (Определить местоположение файла проекта). По умолчанию файл *Key.ini* уже должен быть выбран.

2. Выберите *Open* (Открыть).

3. Установите флажок RSLogix 5000 и нажмите *OK*.

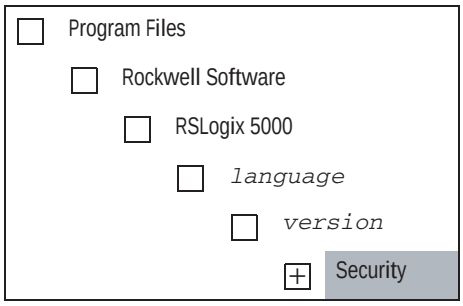
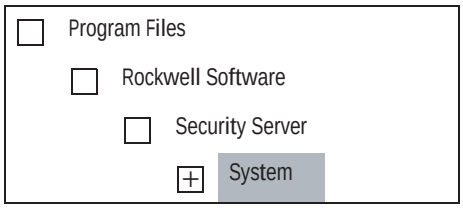


43073

Импорт файла RSLogix5000Security.bak

Файл RSLogix5000Security.bak обеспечивает конфигурацию, необходимую Security Server для работы с программным обеспечением RSLogix 5000.

1. Запустите проводник Security Configuration (Конфигурация защиты).
2. Из меню *File* (Файл) выберите *Import Database* (Импортировать базу данных).
3. Какая версия Security Server вами используется:

Если:	То:
2.00	Смотрите в этой папке: 
2.01	Смотрите в этой папке: 

Где:	Является:
language	языком вашего программного обеспечения. Например, для программного обеспечения на английском языке откройте папку ENU.
version	версией вашего программного обеспечения, например v10

4. Выберите файл *RSLogix5000Security.bak*, а затем выберите *Open* (Открыть).

Задание глобальных действий для пользователей

Глобальные действия представляют собой задачи, не привязанные к конкретному объекту, например, создание нового объекта или обновление микропрограммного обеспечения контроллера. Для программного обеспечения RSLogix 5000 применяются следующие глобальные действия.

Таблица 18.2 Глобальные действия

Чтобы разрешить пользователю:	Предоставьте доступ к следующим действиям:
создавать защиту для любого незащищенного контроллера	Secure Controller (Защита контроллера)
создавать новый проект RSLogix 5000	New Project (Новый проект)
открывать файл .L5K в программе RSLogix 5000, которая создает проект	
переводить проект PLC или SLC в файл .L5K	
использовать программное обеспечение RSLogix 5000 для запуска пакета ControlFLASH и обновления микропрограммного обеспечения контроллера	Update Firmware (Обновление микропрограммного обеспечения)

Используйте следующую рабочую таблицу для записи глобальных действий, выполнение которых вы разрешите каждой из групп пользователей.

Таблица 18.3 Глобальные действия для каждой группы пользователей

Этой группе пользователей:	Требуется доступ к:		
	Security Controller	New Project	Update Firmware

Задание действий проекта для пользователей

Действия проекта позволяют вам выполнять определенные задачи для конкретного проекта или группы проектов.

- Когда вы включаете защиту для проекта RSLogix 5000 или создаете новый проект с включенной защитой, этот проект становится членом группы *New RSLogix 5000 Resources* (Новые ресурсы RSLogix 5000).
  - Пользователям, которые работают с проектами этой группы, требуется иметь соответствующий доступ.
  - Мы рекомендуем предоставлять *Full Access* (Полный доступ) всем пользователям, имеющим доступ к созданию проекта.

- Чтобы настроить доступ для проекта, вытащите его из группы *New RSLogix 5000 Resources* и присвойте привилегии, относящиеся к данному конкретному проекту.

Для защищенного проекта или группы проектов RSLogix 5000 применяются следующие действия.

Таблица 18.4 Действия проекта

Чтобы разрешить пользователю:	А также:	А также:	Предоставьте доступ к этому действию:
• открывать проект в режиме оффлайн	→	→	View Project (Просмотр проекта)
• копировать компоненты из проекта • экспортировать теги проекта	выходить в режим онлайн и контролировать работу проекта	→	Go Online (Выход в режим онлайн)
		• сохранять проект • сохранять проект в другом файле .ACD • открывать более раннюю версию проекта • сжимать проект • экспортировать проект • загружать или выгружать проект • изменять режим контроллера • изменять путь к контроллеру • распечатывать отчет • сбрасывать ошибки • изменять время на часах • создавать, удалять, редактировать и выполнять тренд • изменять конфигурацию модуля ввода/вывода • изменять конфигурацию инструкции MSG • вводить, включать, выключать и удалять силы • изменять значения тегов • обновлять микропрограммное обеспечение	Maintain Project (Сопровождение проекта)
выполнять все действия, доступные для программного обеспеченияRSLogix 5000, *кроме* удаления защиты защищенного контроллера	→	→	Full Access (Полный доступ)
удалять защиту защищенного контроллера	→	→	Full Access и Unsecure Controller (Удаление защиты контроллера)
обновлять микропрограммное обеспечение контроллера	→	→	Update Firmware (Обновление микропрограммного обеспечения)

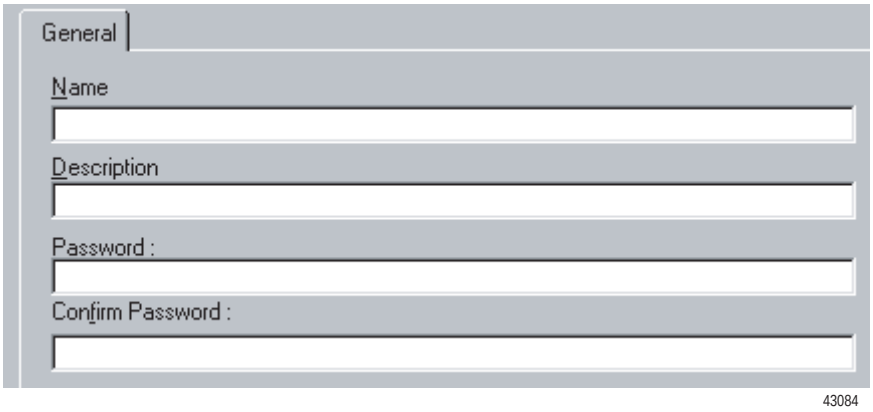
Используйте рабочую таблицу на стр. 18-19 для записи действий проекта, выполнение которых вы разрешите каждому пользователю или группе пользователей.

Таблица 18.5 Действия проекта для проектов, входящих в группу New RSLogix 5000 Resources, а также для индивидуальных проектов

Для этого проекта или группы проектов:	Этому пользователю или группе пользователей:	Требуется доступ к:					
		View Project	Go Online	Maintain Project	Full Access	Unsecure Controller	Update Firmware
New RSLogix 5000 Resources							
New RSLogix 5000 Resources							
New RSLogix 5000 Resources							
New RSLogix 5000 Resources							

Добавление пользователей

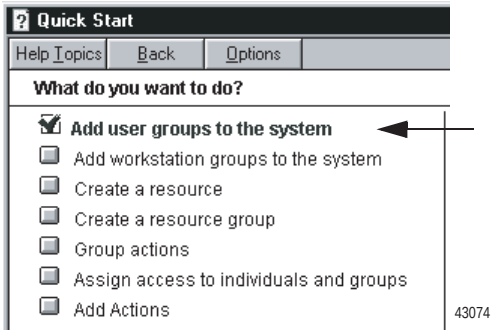
1. Нажмите правую кнопку мыши и выберите *New* (Новый).



2. Введите информацию о пользователе и нажмите *OK*

Добавление групп пользователей

Создание группы позволяет вам управлять многочисленными пользователями, которым требуются одинаковые привилегии.



1. Из меню *Help* (Справка) выберите *Quick Start* (Быстрое начало).

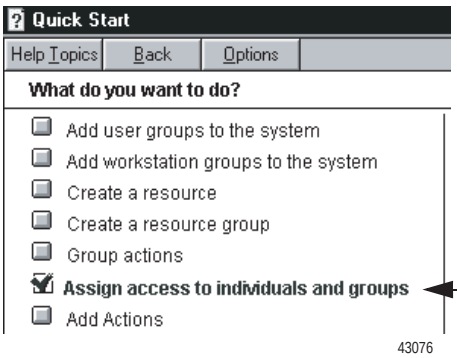
2. Последовательно выполняйте шаги для этой задачи.

Назначение глобального доступа к программному обеспечению RSLogix 5000

Чтобы разрешить пользователям выполнять глобальные действия:

1. В проводнике Configuration (Конфигурирование) выберите группу *RSLOGIX 5000*.

2. Из меню *Help* (Справка) выберите *Quick Start* (Быстрое начало).



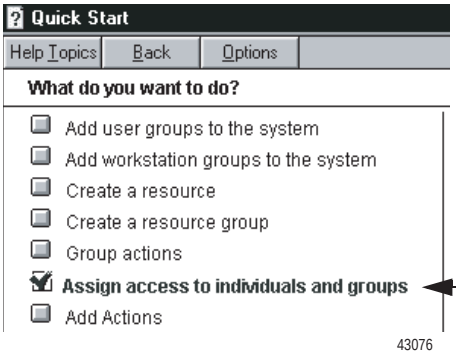
3. Последовательно выполняйте шаги для этой задачи. Назначьте действия, которые вы записали в Таблицу 18.3 на стр. 18-16.

Назначение действий проекта новым проектам RSLogix 5000

Чтобы разрешить пользователям выполнять действия над проектами, находящимися в группе New RSLogix 5000 Resources (Новые ресурсы RSLogix 5000):

1. В проводнике Configuration (Конфигурирование) выберите группу *New RSLogix 5000 Resources.*

2. Из меню *Help* (Справка) выберите *Quick Start* (Быстрое начало).

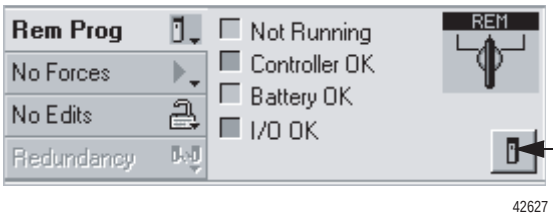


3. Последовательно выполняйте шаги для этой задачи. Назначьте действия, которые вы записали в Таблицу 18.5 на стр. 18-19.

Создание защиты для проекта RSLogix 5000

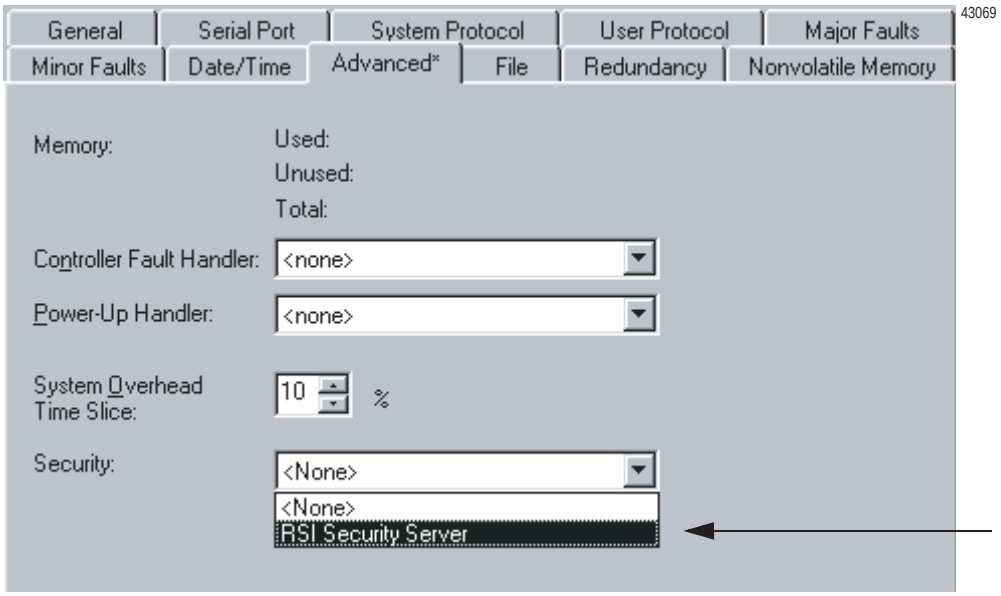
Для новых проектов опция защиты доступна при создании проекта. Чтобы разрешить программе Security Server обеспечить защиту существующего проекта, активируйте защиту для этого проекта.

1. Откройте проект RSLogix 5000.



2. Нажмите на кнопку свойств контроллера.

3. Нажмите на закладку *Advanced* (Расширенные).



4. Выберите *RSI Security Server*.

5. Нажмите *OK*, а затем *Yes*.

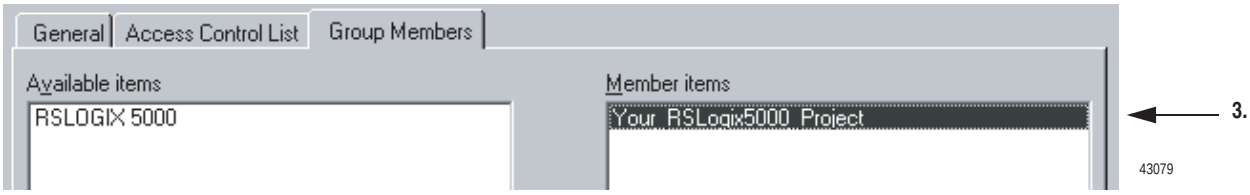
В программе Security Server проект является членом группы *New RSLogix 5000 Resources*. Если программа Security Server уже открыта, то из меню *View* (Вид) выберите *Refres*h (Обновить).

Назначение доступа к проекту RSLogix 5000

Пока проект находится в группе New RSLogix 5000 Resources, перечень управления доступом этой группы определяет действия, которые пользователь может выполнять с проектом. Чтобы настроить доступ для проекта, вытащите его из группы и назначьте конкретные действия:

1. В проводнике Configuration (Конфигурирование) выберите группу *New RSLogix 5000 Resources.*

2. Нажмите на закладку *Group Members* (Члены группы).

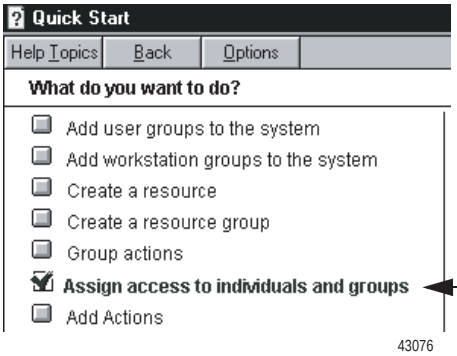


3. В перечне *Member items* (Члены) выберите проект и нажмите на кнопку <<.

4. Выберите *Apply* (Применить).

5. В проводнике Configuration выберите проект.

6. Из меню *Help* (Справка) выберите *Quick Start* (Быстрое начало).



7. Последовательно выполняйте шаги для этой задачи. Назначьте действия, которые вы записали в Таблицу 18.5 на стр. 18-19.

Обновление программного обеспечения RSLogix 5000 при необходимости

Если проект RSLogix 5000 открыт, а изменения, сделанные в программе RSI Security Server, влияют на этот проект, обновите программное обеспечение RSLogix 5000:

Из меню *Tools* (Сервис) выберите *Security* (Защита) => *Refresh Privileges* (Обновить привилегии).

Примечания:

Получение информации о памяти контроллера

Когда использовать эту главу

Используйте эту главу для получения информации о памяти вашего контроллера Logix5000.

Определение требуемой информации о памяти

В зависимости от типа вашего контроллера память контроллера может быть подразделена на несколько областей:

Если у вас этот контроллер:	То он хранит это:	В этой памяти:
ControlLogix	теги ввода-вывода	память ввода-вывода
	производимые теги	
	потребляемые теги	
	обмен данными посредством инструкций Message (MSG)	
	обмен данными с рабочими станциями	
	обмен данными с тегами опроса (OPC/DDE), использующими программное обеспечение RSLinx[1]	
	теги, отличные от тегов ввода-вывода, производимых и потребляемых тегов	память для логики и данных[2]
	процедуры логики	
	обмен данными с тегами опроса (OPC/DDE), использующими программное обеспечение RSLinx[1]	
• CompactLogix • FlexLogix • DriveLogix • SoftLogix5800	У этих контроллеров память *не* разделена. Все элементы хранятся в одной общей области памяти.	

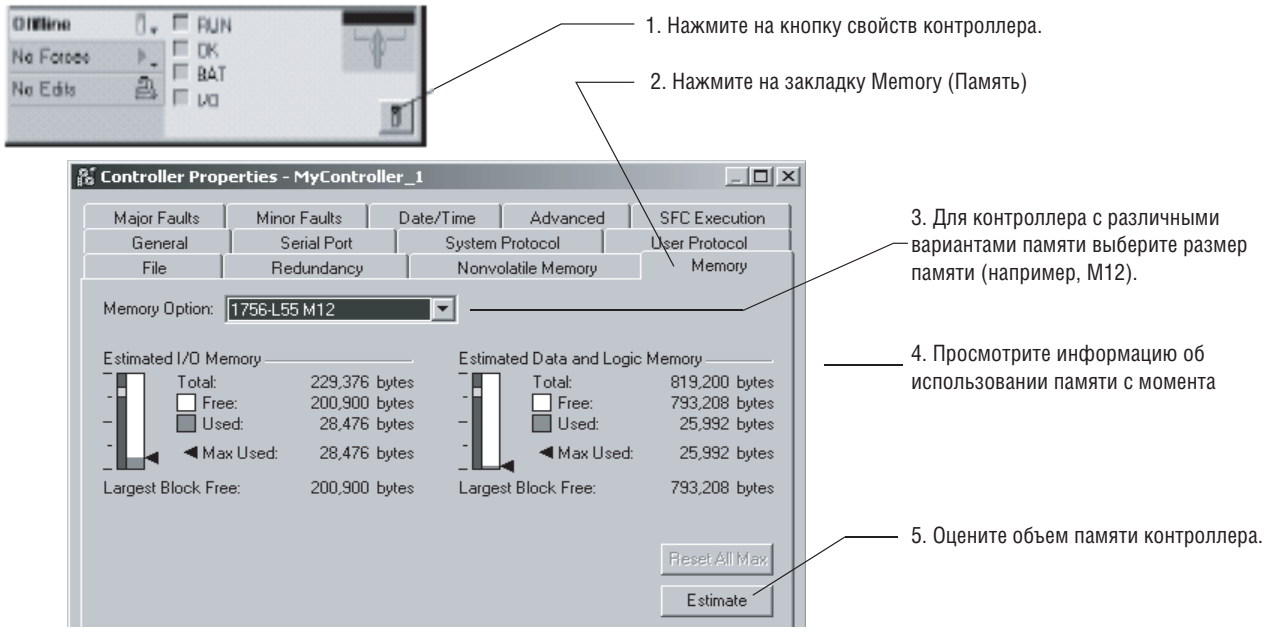
[1]Для обмена данными с тегами опроса контроллер использует как память ввода-вывода, так и память для логики и данных.

[2]Контроллеры 1756-L55M16 имеют дополнительный раздел памяти для логики.

Оценка информации об использовании памяти в режиме оффлайн

Чтобы оценить, сколько памяти контроллера требуется для вашего проекта, используйте закладку *Memory* (Память) в диалоговом окне свойств контроллера. Здесь для каждой области памяти вашего контроллера вы можете оценить количество байтов:

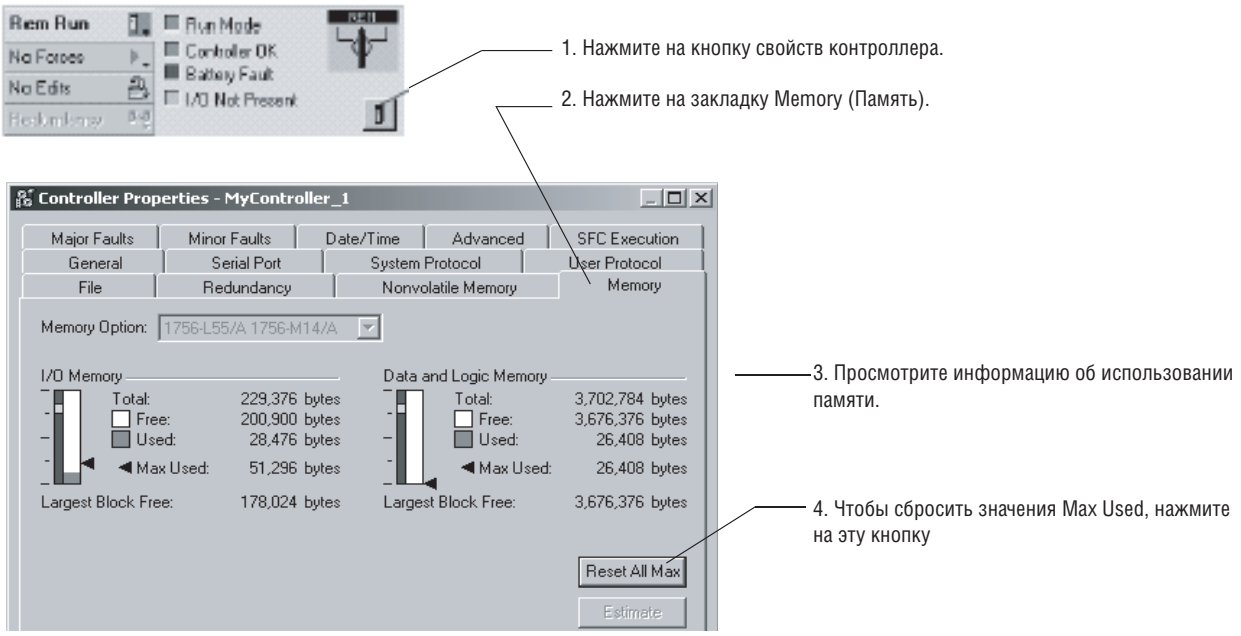
- свободной (неиспользуемой) памяти
- используемой памяти
- максимального свободного непрерывного блока памяти



Просмотр информации об использовании памяти в режиме выполнения

Когда контроллер находится в режиме онлайн, в закладке *Memory* (Память) отображается фактическое использование памяти контроллера. Когда контроллер работает, он использует дополнительную память для обмена данными. Объем памяти, который необходим контроллеру, варьируется в зависимости от состояния обмена данных.

Закладка *Memory* контроллера содержит элемент *Max Used* (Максимум использования) для каждого типа памяти. Значения *Max Used* показывают пик использования памяти во время обмена данными.



Написание логики для получения информации о памяти

Чтобы использовать логику для получения информации о памяти для контроллера:

- Получите информацию о памяти из контроллера
- Выберите требуемую информацию о памяти
- Преобразуйте значения INT в DINT

Получение информации о памяти из контроллера

Чтобы получить информацию о памяти из контроллера, выполните инструкцию Message (MSG), которая сконфигурирована следующим образом:

В этой закладке:	Для этого элемента:	Введите или выберите:	Что означает:
Configuration (Конфигурация)	Message Type (тип сообщения)	CIP Generic	Выполнение команды Control and Information Protocol (Протокол управления и информации)
	Service Type (тип сервиса)	Custom (специальный)	Создание сообщения CIP Generic, отсутствующего в выпадающем списке
	Service Code (код сервиса)	3	Считывание определенной информации о контроллере (сервис GetAttributeList)
	Class (класс)	72	Получение информации от объекта пользовательской памяти.
	Instance (экземпляр)	1	Этот объект содержит только один экземпляр.
	Attribute (атрибут)	0	Нулевое значение
	Source Element (исходный элемент)	исходный массив *source_array* типа SINT[12]	
		В этом элементе:	**Введите:**	**Что означает:**
		source_array[0]	5	Получение 5 атрибутов
		source_array[1]	0	Нулевое значение
		source_array[2]	1	Получение объема свободной памяти
		source_array[3]	0	Нулевое значение
		source_array[4]	2	Получение общего объема памяти
		source_array[5]	0	Нулевое значение
		source_array[6]	5	Получение размера максимального непрерывного блока дополнительной свободной памяти логики
		source_array[7]	0	Нулевое значение
		source_array[8]	6	Получение размера максимального непрерывного блока свободной памяти ввода-вывода
		source_array[9]	0	Нулевое значение
		source_array[10]	7	Получение размера максимального непрерывного блока свободной памяти логики и данных
		source_array[11]	0	Нулевое значение
	Source Length (исходная длина)	12	Запись 12 байтов (12 значений SINT)
	Destination (адресат)	массив *INT_array* типа INT[29]	
Communication (Связь)	Path (путь)	1, *slot_number_of_controller* (íîìåð ñëîòà êîíòðîëëåðà)	

Выбор требуемой информации о памяти

Инструкция MSG возвращает следующую информацию в *INT_array* (тег-приемник MSG):

ВАЖНО

- Контроллер возвращает значения в виде 32-битовых слов. Чтобы получить значение в байтах, умножьте его на 4.
- Если ваш контроллер не подразделяет память, то значения отображаются как в памяти ввода-вывода.
- Для контроллера 1756-L55M16 инструкция MSG возвращает два значения для каждой категории памяти логики. Чтобы определить размер свободной или общей памяти логики контроллера 1756-L55M16, сложите оба значения для соответствующей категории.

Если вам требуется:	То скопируйте эти элементы массива:	Описание:
объем свободной памяти ввода-вывода (32-битовые слова)	*INT_array[3]*	младшие16 бит 32-битового значения
	INT_array[4]	старшие 16 бит 32-битового значения
объем свободной памяти логики и данных (32-битовые слова)	*INT_array[5]*	младшие16 бит 32-битового значения
	INT_array[6]	старшие 16 бит 32-битового значения
только для контроллеров 1756-L55M16 – объем дополнительной свободной памяти логики (32-битовые слова)	*INT_array[7]*	младшие16 бит 32-битового значения
	INT_array[8]	старшие 16 бит 32-битового значения
общий объем памяти ввода-вывода (32-битовые слова)	*INT_array[11]*	младшие16 бит 32-битового значения
	INT_array[12]	старшие 16 бит 32-битового значения
общий объем памяти для логики и данных (32-битовые слова)	*INT_array[13]*	младшие16 бит 32-битового значения
	INT_array[14]	старшие 16 бит 32-битового значения
только для контроллеров 1756-L55M16 – дополнительная память логики (32-битовые слова)	*INT_array[15]*	младшие16 бит 32-битового значения
	INT_array[16]	старшие 16 бит 32-битового значения
только для контроллеров 1756-L55M16 – максимальный непрерывный блок дополнительной свободной памяти логики (32-битовые слова)	*INT_array[19]*	младшие16 бит 32-битового значения
	INT_array[20]	старшие 16 бит 32-битового значения
максимальный непрерывный блок памяти ввода-вывода (32-битовые слова)	*INT_array[23]*	младшие16 бит 32-битового значения
	INT_array[24]	старшие 16 бит 32-битового значения
максимальный непрерывный блок свободной памяти логики и данных (32-битовые слова)	*INT_array[27]*	младшие16 бит 32-битового значения
	INT_array[28]	старшие 16 бит 32-битового значения

Преобразование INT в DINT

Инструкция MSG возвращает каждое значение памяти в виде двух отдельных значений INT.

- Первое значение INT представляет собой младшие 16 бит соответствующего значения.
- Второе значение INT представляет собой старшие 16 бит этого значения.

Чтобы преобразовать отдельные значения INT в одно удобное для использования значение, используйте инструкцию Copy (COP), где:

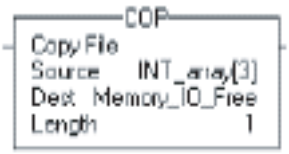
В этом операнде:	Задайте:	Что означает:
Source	первое значение INT из пары элементов (младшие 16 бит)	Начать с младших 16 бит
Destination	тег DINT, в котором будет храниться 32-битовое значение	Копировать значение в тег DINT
Length	1	Однократно копировать количество байтов в типе данных Destination. В этом случае инструкция копирует 4 байта (32 бита), при этом младшие и старшие 16 бит объединяются в одно 32-битовое значение.

В следующем примере инструкция COP производит 32-битовое значение, соответствующее объему свободной памяти ввода-вывода, в виде 32-битовых слов.

ПРИМЕР

Преобразование значений INT в DINT

- Элемент 3 массива *INT_array* – это младшие 16 бит объема свободной памяти ввода-вывода. Элемент 4 – старшие 16 бит.
- *Memory_IO_Free* – это тег DINT (32 бита), в котором будет храниться значение объема свободной памяти ввода-вывода.
- Чтобы скопировать все 32 бита, задайте Length (длину), равную 1. Это указывает инструкции, что надо 1 раз скопировать размер Destination (адресата) (32 бита). Команда скопирует и элемент 3 (16 бит), и элемент 4 (16 бит), и поместит 32-битовый результат в тег *Memory_IO_Free*.



Управление многочисленными сообщениями

Назначение

В этом приложении описывается, как использовать релейную логику для последовательной отправки групп инструкций Message (MSG). При этом инструкции смогут упорядоченно входить в очередь сообщений и выходить из нее.

Когда использовать это приложение

Используйте это приложение, если вам необходимо управлять выполнением большого количества инструкций MSG.

- Для обработки каждая инструкция MSG должна войти в очередь сообщений.
- Очередь вмещает 16 инструкций MSG.
- Если одновременно разрешены более 16 инструкций MSG, в очереди может не хватить места, когда будет разрешена еще одна инструкция MSG.
- Если это произойдет, инструкции MSG придется ждать до тех пор, пока в очереди не появится свободное место, чтобы контроллер смог обработать эту инструкцию MSG. При каждом последующем сканировании MSG, инструкция проверяет очередь на наличие свободного места.

Логика диспетчера сообщений, приведенная в этом приложении, позволит вам управлять количеством одновременно разрешенных инструкций MSG и поочередно разрешать последующие инструкции MSG. Таким образом инструкции MSG будут входить в очередь и выходить из нее упорядоченно, и им не придется ждать свободного места в очереди, чтобы стать доступными.

Как использовать это приложение

Описываемая в этом приложении логика диспетчера сообщений посылает три группы инструкций MSG.

- Для наглядности каждая группа в этом примере содержит только две инструкции MSG.
- Для вашего проекта используйте большее количество инструкций MSG в каждой группе, например, 5.
- Используйте столько групп, сколько необходимо, чтобы в них вошли все ваши инструкции MSG.

Тег *Msg_Group* управляет разрешением каждой инструкции MSG.

- Этот тег использует тип данных DINT.
- Каждый бит тега соответствует одной группе инструкций MSG.
- Например, *Msg_Group.0* разрешает и запрещает первую группу инструкций MSG (группа 0).

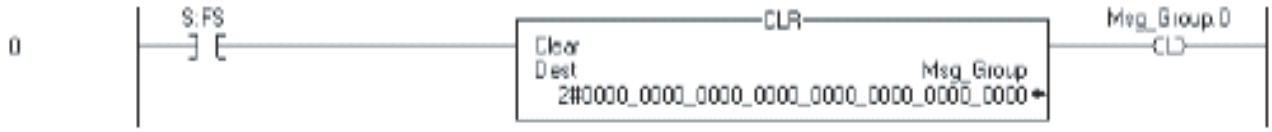
Логика диспетчера сообщений

Инициализация логики

Если S:FS = 1 (первое сканирование), то инструкции MSG инициализируются:

Msg_Group = 0, что запрещает все инструкции MSG.

Msg_Group.0 = 1, что разрешает первую группу инструкций MSG.



Перезапуск последовательности при необходимости

Если инструкции MSG в группе 2 (последняя группа) в данный момент разрешены (*Msg_Group.2* = 1)

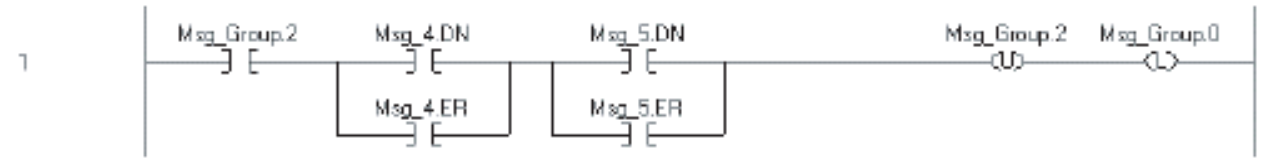
И *Msg_4* выполнено или ошибочно,

А также *Msg_5* выполнено или ошибочно,

То перезапускается последовательность инструкций MSG, начиная с первой группы:

Msg_Group.2 = 0. Это запрещает последнюю группу инструкций MSG.

Msg_Group.0 = 1. Это разрешает первую группу инструкций MSG.



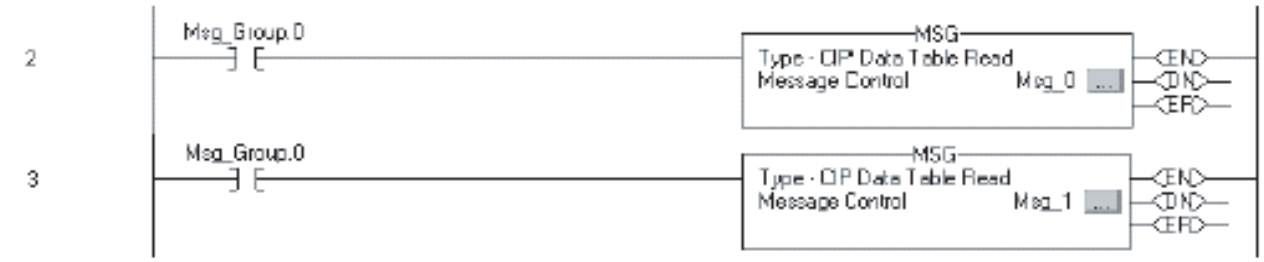
Отправка первой группы инструкций MSG

Если *Msg_Group.0* меняется с 0 на 1, то

Отправляется *Msg_0*.

Отправляется *Msg_1*.

Поскольку инструкция MSG является инструкцией перехода, она выполняется лишь в том случае, когда входное условие цепочки меняется с «ложно» на «истинно».

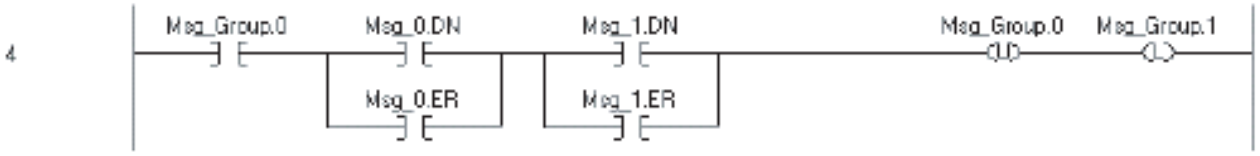


Разрешение следующей группы инструкций MSG

Если инструкции MSG в группе 0 в данный момент разрешены (*Msg_Group.0* = 1)
И *Msg_0* выполнено или ошибочно,
А также *Msg_1* выполнено или ошибочно,
То

Msg_Group.0 = 0. Это запрещает текущую группу инструкций MSG.

Msg_Group.1 = 1. Это разрешает следующую группу инструкций MSG.

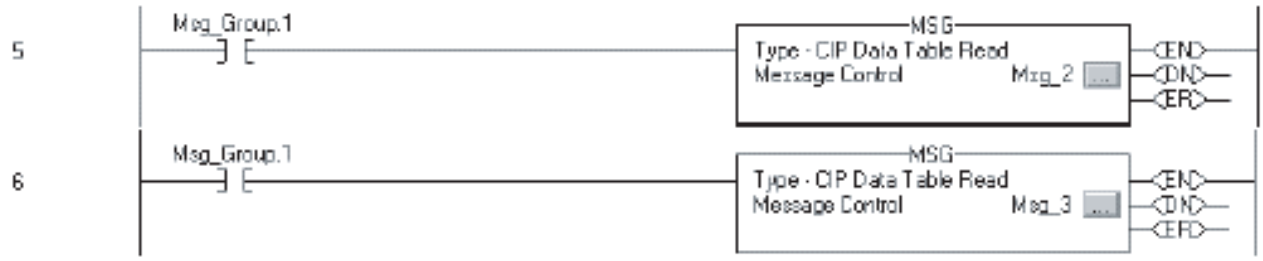


Отправка следующей группы инструкций MSG

Если *Msg_Group.1* меняется с 0 на 1, то

Отправляется *Msg_2*.

Отправляется *Msg_3*.



Разрешение следующей группы инструкций MSG

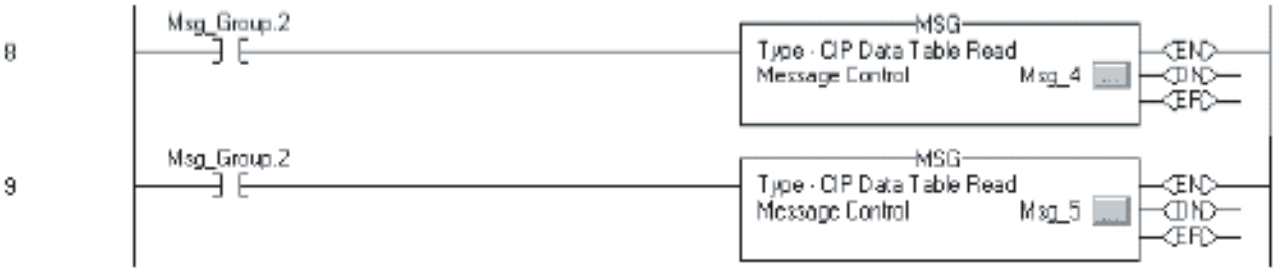
Если инструкции MSG в группе 1 в данный момент разрешены (*Msg_Group.1* = 1)
И *Msg_2* выполнено или ошибочно,
А также *Msg_3* выполнено или ошибочно,
То

Msg_Group.1 = 0. Это запрещает текущую группу инструкций MSG.
Msg_Group.2 = 1. Это разрешает следующую группу инструкций MSG.

Отправка следующей группы инструкций MSG

Если *Msg_Group.1* меняется с 0 на 1, то

Отправляется *Msg_2*.
Отправляется *Msg_3*.



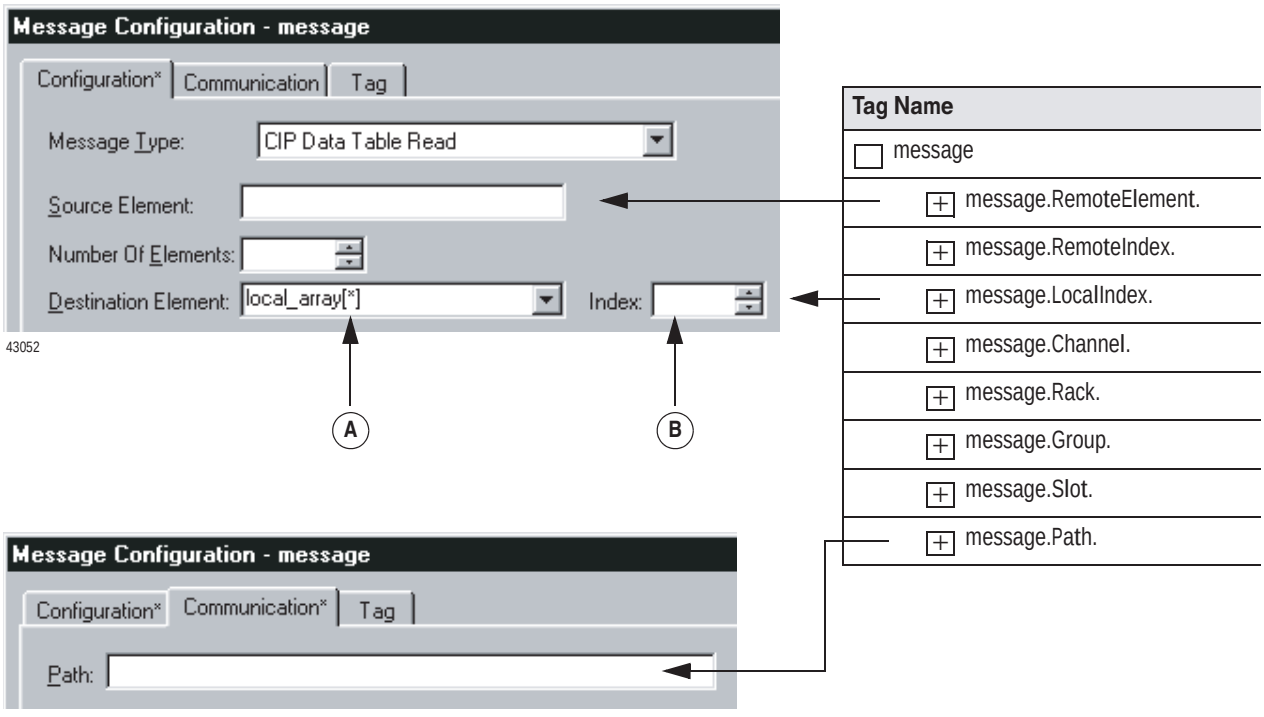
Отправка сообщения нескольким контроллерам

Используйте следующую процедуру, чтобы спрограммировать обмен данными между одной инструкцией сообщения и несколькими контроллерами. Чтобы переконфигурировать инструкцию MSG в процессе выполнения, запишите новые значения в члены типа данных MESSAGE.

ВАЖНО

В типе данных MESSAGE член RemoteElement хранит имя тега или адрес данных в контроллере, по которому будет получено данное сообщение.

Если данное сообщение:	То RemoteElement представляет собой:
считывает данные	Source Element (исходный элемент)
записывает данные	Destination Element (целевой элемент)



A. Если вы используете «звездочку» [*] для обозначения номера элемента массива, значение в поле **B** указывает номер данного элемента.

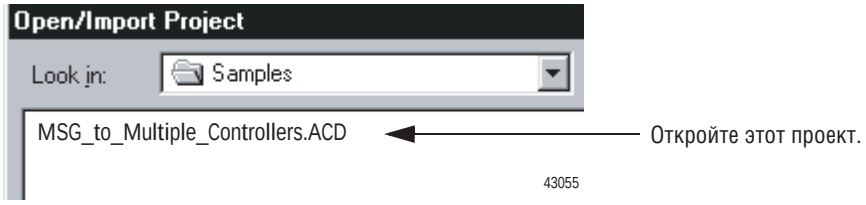
B. Поле *Index* (Индекс) доступно только в том случае, когда вы используете «звездочку» [*] в поле *Source Element* (Исходный элемент) или *Destination Element* (Целевой элемент). Инструкция заменяет «звездочку» [*] значением из поля *Index*.

Чтобы послать сообщение нескольким контроллерам:

- Создайте конфигурацию ввода-вывода
- Задайте ваши элементы-источники и приемники
- Создайте тип данных MESSAGE_CONFIGURATION
- Создайте массив конфигурации
- Определите размер локального массива
- Загрузите свойства сообщения для контроллера
- Сконфигурируйте сообщение
- Перейдите к следующему контроллеру
- Перезапустите последовательность.

СОВЕТ

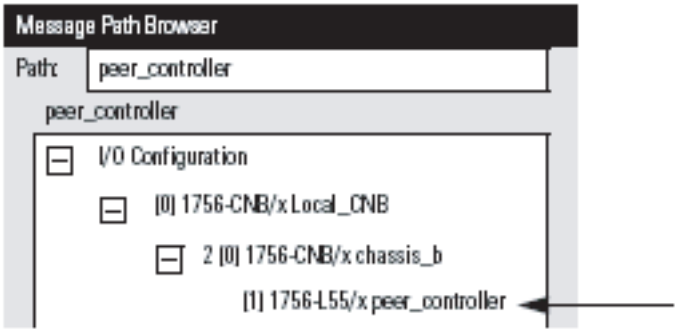
Чтобы скопировать вышеуказанные компоненты из типового проекта, откройте папку *...\RSLogix 5000\Projects\Samples*.



Создание конфигурации ввода-вывода

Хотя это и не обязательно, мы рекомендуем вам добавить модули связи и удаленные контроллеры к конфигурации ввода-вывода контроллера. Это упростит задание пути к каждому удаленному контроллеру.

Например, после того как вы добавите локальный модуль связи, удаленный модуль связи и контроллер назначения, кнопка *Browse* (Просмотр) позволит вам выбирать пункт назначения.



Задание элементов-источников и приемников

В этой процедуре в массиве хранятся данные, которые считываются с каждого удаленного контроллера или записываются на каждый удаленный контроллер. Каждый элемент массива соответствует своему удаленному контроллеру.

1. Используйте следующую рабочую таблицу, чтобы организовать имена тегов в локальных и удаленных контроллерах.

Имя удаленного контроллера:	Тег или адрес данных в удаленном контроллере:	Тег в этом контроллере:
		local_array[0]
		local_array[1]
		local_array[2]
		local_array[3]

2. Создайте тег *local_array*, который будет хранить данные в этом контроллере.

Имя тега	Тип
local_array	*data_type* [length] где: *data_type* – тип данных, отправляемых или получаемых сообщением, например, DINT, REAL или STRING *length* – количество элементов в данном локальном массиве.

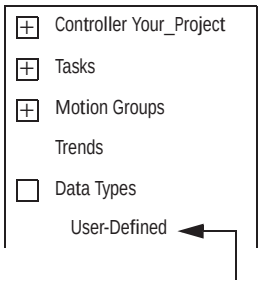
Создание типа данных MESSAGE_CONFIGURATION

В этой процедуре вы создаете пользовательский тип данных для хранения переменных конфигурации для сообщения, отправляемого на каждый контроллер.

- Некоторые обязательные члены этого типа данных используют строковый тип данных.
- По умолчанию тип данных STRING (строка) сохраняет 82 символа.
- Если ваши пути, имена удаленных тегов или адреса включают менее 82 символов, вы можете создать новый тип строки, в котором будет храниться меньшее количество символов. Это позволит сэкономить память.
- Чтобы создать новый тип строки, выберите *File* (Файл) => *New Component* (Новый компонент) => *String Type...* (Строковый тип)
- Если вы создадите новый тип строки, используйте его вместо типа данных STRING в этой процедуре.

Чтобы сохранять переменные конфигурации для сообщения, отправляемого на каждый контроллер, создайте следующий пользовательский тип данных.

Для создания нового типа данных:



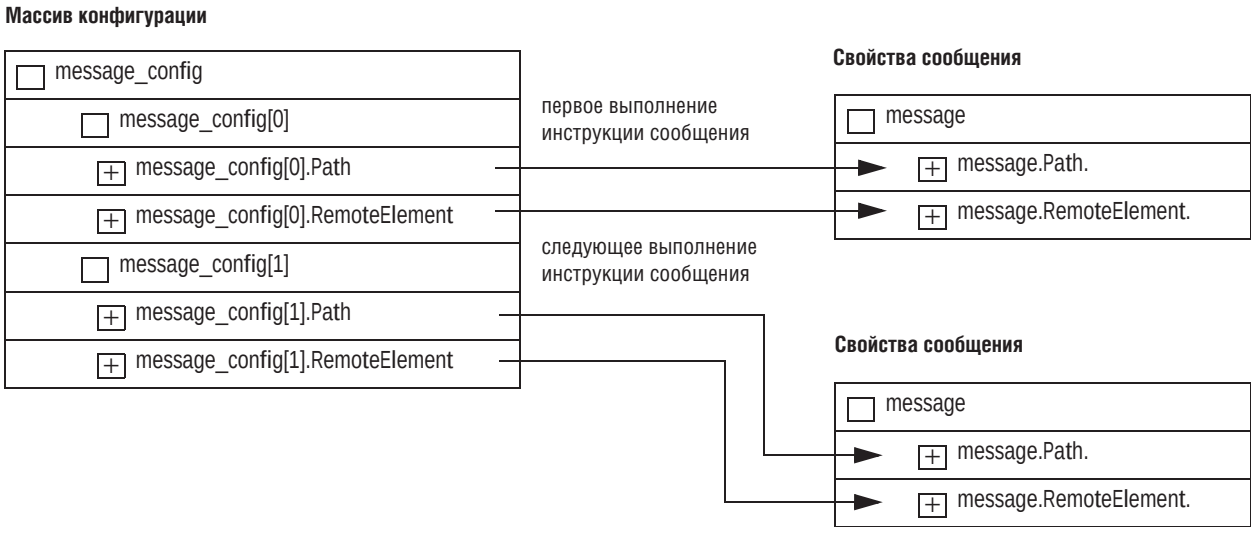
Щелкните правой кнопкой мыши и выберите New Data Type (Новый тип данных)

Тип данных: MESSAGE_CONFIGURATION			
Имя	MESSAGE_CONFIGURATION		
Описание	Свойства конфигурации для сообщения, отправляемого на другой контроллер		
Члены			
Имя	**Тип данных**	**Стиль**	**Описание**
Path	STRING		
RemoteElement	STRING		

Создание массива конфигурации

В этой процедуре вы сохраняете свойства конфигурации для каждого контроллера в массиве. Перед каждым выполнением инструкции MSG ваша логика загружает в нее новые свойства. При этом сообщение отсылается на другой контроллер.

Рисунок B.1 Загрузка новых свойств конфигурации в инструкцию MSG



Шаги:

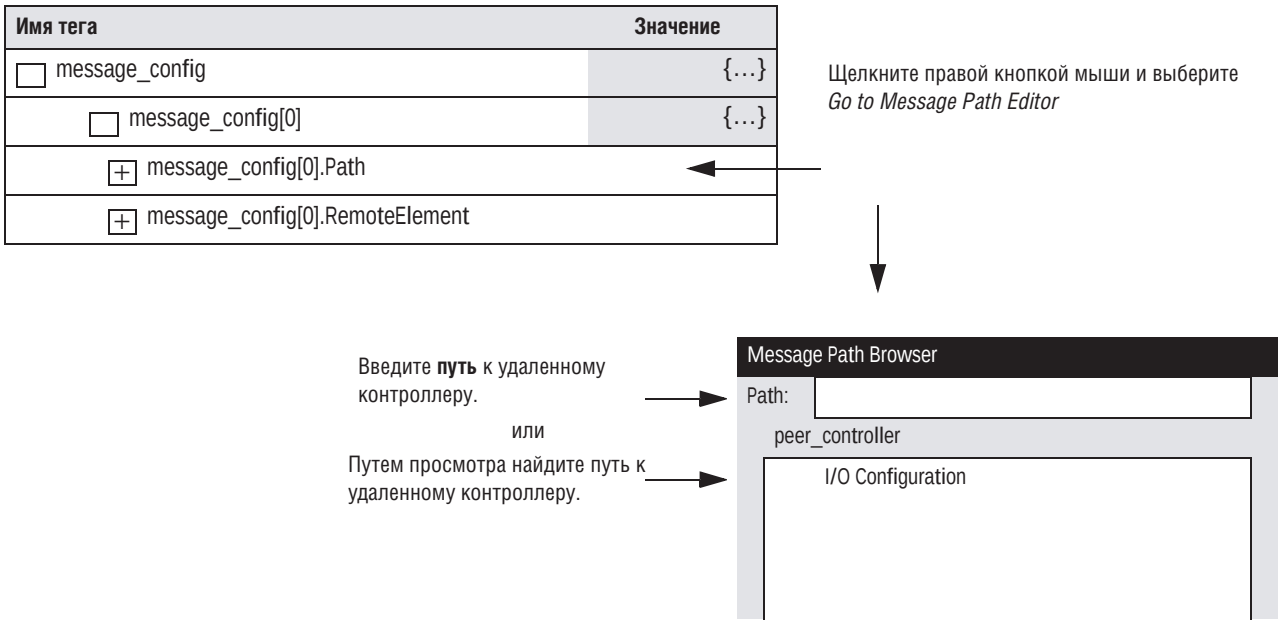
1. Чтобы сохранить свойства конфигурации для сообщения, создайте следующий массив:

Имя тега	Тип	Область видимости
message_config	MESSAGE_CONFIGURATION[*number*]	любая

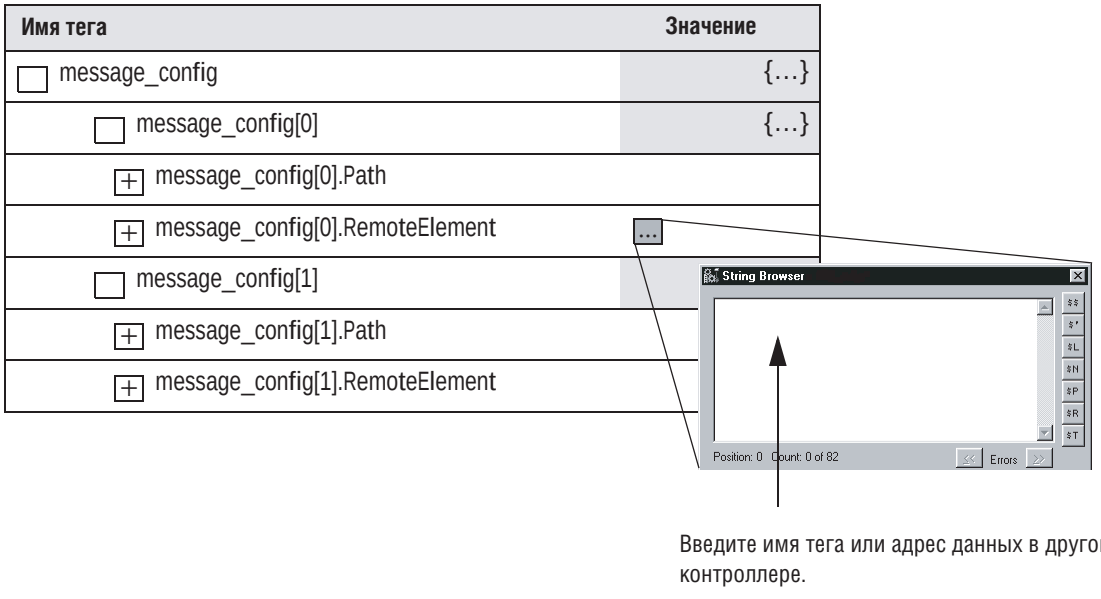
где:

number – количество контроллеров, на которые должно быть отправлено сообщение.

2. В массив *message_config* введите **путь** к первому контроллеру, который получит данное сообщение.



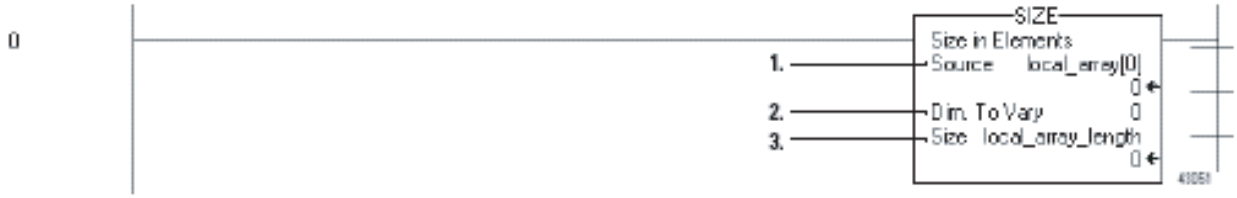
3. В массив *message_config* введите имя тега или адрес данных в первом контроллере, который получит данное сообщение.



4. Введите путь и удаленный элемент для каждого дополнительного контроллера:

Имя тега	Значение
☐ message_config	{…}
☐ message_config[0]	{…}
⊞ message_config[0].Path	
⊞ message_config[0].RemoteElement	
☐ message_config[1]	{…}
⊞ message_config[1].Path	←
⊞ message_config[1].RemoteElement	←

Определение размера локального массива



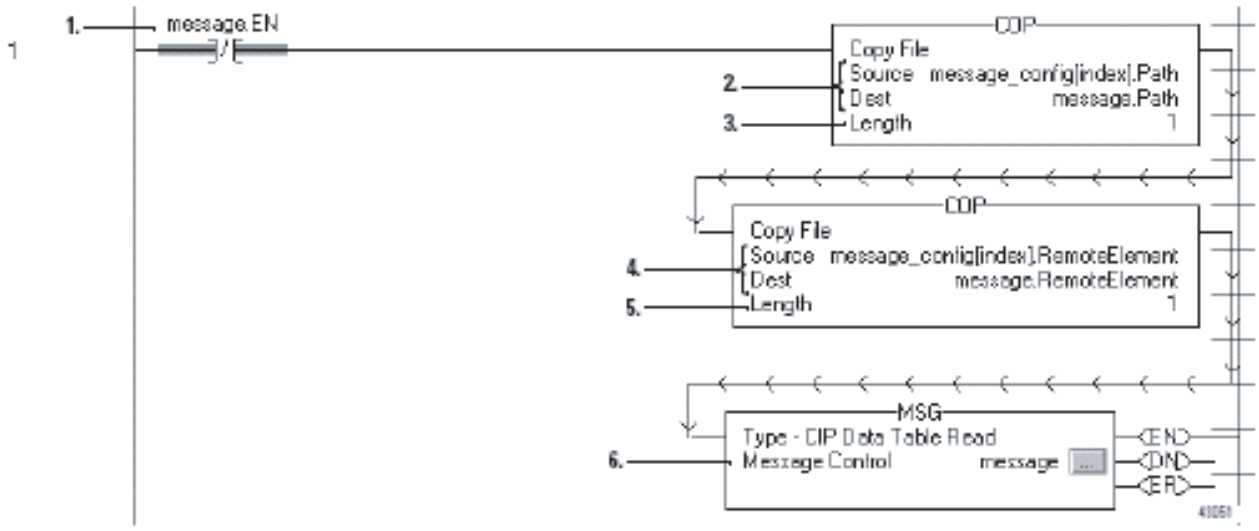
1. Инструкция SIZE подсчитывает количество элементов в массиве *local_array*.

2. Инструкция SIZE подсчитывает количество элементов в Dimension 0 (измерении 0) массива. В данном случае это единственное измерение.

3. Тег *Local_array_length* сохраняет размер (количество элементов) массива *local_array*. Это значение сообщает следующей цепочке об отправке сообщения всем контроллерам и указывает ей начать снова с первого контроллера.

Имя тега	Тип
local_array_length	DINT

Загрузка свойств сообщения для контроллера



1. Эта инструкция XIO обусловливает непрерывную отправку сообщения данной цепочкой.

Имя тега	Тип	Область видимости
message	MESSAGE	контроллер

2. Инструкция COP загружает путь для данного сообщения. Значение *index* определяет, какой элемент загружается этой инструкцией из *message_config*. См. рис. B.1 на стр. B-6.

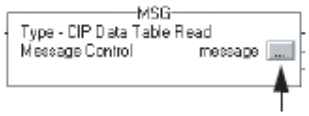
Имя тега	Тип	Область видимости
index	DINT	любая

3. Инструкция загружает один элемент из *message_config*.

4. Инструкция COP загружает имя тега или адрес данных в контроллере-получателе сообщения. Значение *index* определяет, какой элемент загружается этой инструкцией из *message_config*. См. рис. B.1 на стр. B-6.

5. Инструкция загружает один элемент из *message_config*.

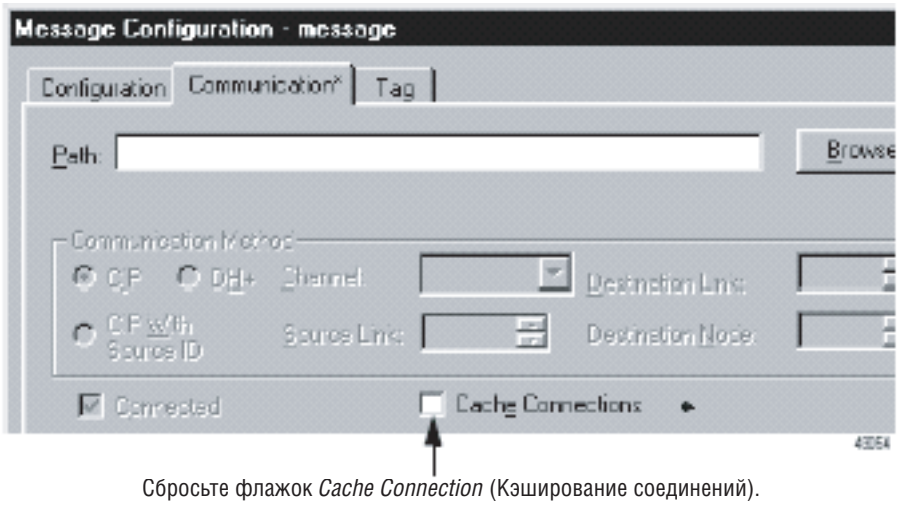
6. Инструкция MSG



Конфигурирование сообщения

Хотя ваша логика управляет удаленным элементом и путем для сообщения, диалоговое окно Message Properties (Свойства сообщения) требует первоначального конфигурирования.

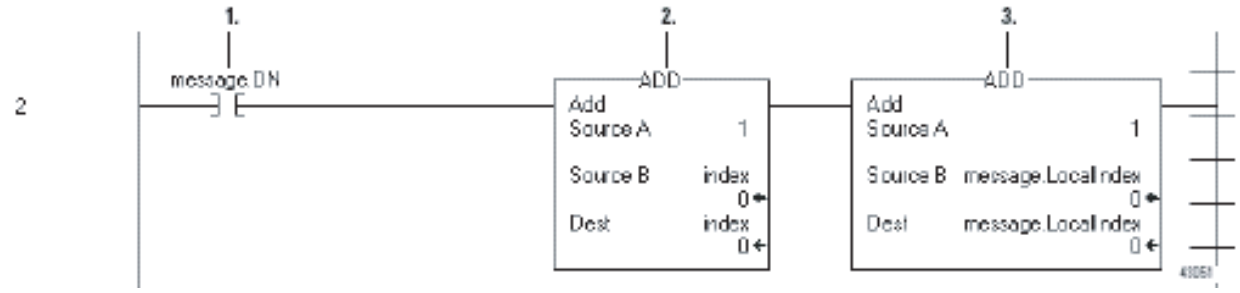
ВАЖНО



Сбросьте флажок *Cache Connection* (Кэширование соединений).

В этой закладке:	Если вы хотите:	Для этого элемента:	Введите или выберите:
Configuration (Конфигурация)	считывать (получать) данные с других контроллеров	Message Type (тип сообщения)	тип чтения, соответствующий другим контроллерам
		Source Element (исходный элемент)	тег или адрес, содержащий данные в первом контроллере
		Number Of Elements (количество элементов)	1
		Destination Tag (целевой тег)	local_array[*]
		Index (индекс)	0
	записывать (отправлять) данные на другие контроллеры	Message Type (тип сообщения)	тип записи, соответствующий другим контроллерам
		Source Tag (исходный тег)	local_array[*]
		Index (индекс)	0
		Number Of Elements (количество элементов)	1
		Destination Element (целевой элемент)	тег или адрес, содержащий данные в первом контроллере
Communication (Связь)	→	Path (путь)	путь к первому контроллеру
	→	Cache Connections (Кэширование соединений)	Сбросьте флажок *Cache Connection* (Кэширование соединений). Поскольку эта процедура постоянно изменяет путь сообщения, более эффективным будет сбросить этот флажок.

Переход к следующему контроллеру

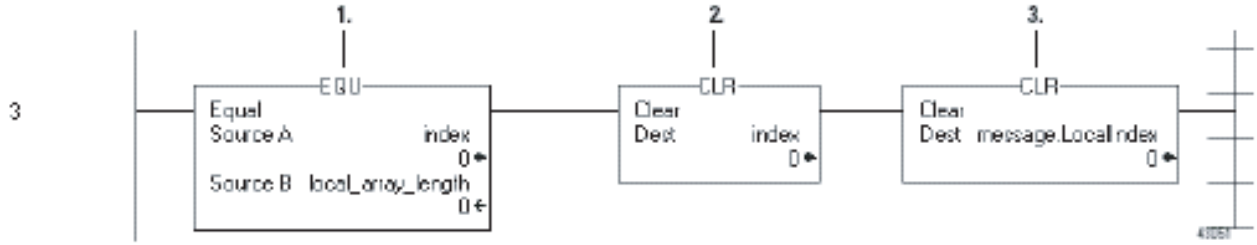


1. После того, как инструкция MSG отправит сообщение...

2. Эта инструкция ADD дискретно увеличивает значение *index*. Это позволяет логике загрузить в инструкцию MSG свойства конфигурации для следующего контроллера.

3. Эта инструкция ADD дискретно увеличивает член *LocalIndex* инструкции MSG. Это позволяет логике загрузить соответствующее значение из следующего контроллера в следующий элемент массива *local_array*.

Перезапуск последовательности



1. Когда индекс равен *local_array_length*, это означает, что контроллер отправил сообщение всем остальным контроллерам.

2. Эта инструкция CLR устанавливает *index* равным 0. Это позволяет логике загрузить в инструкцию MSG свойства конфигурации для первого контроллера и начать последовательность сообщений снова.

3. Эта инструкция CLR устанавливает член *LocalIndex* инструкции MSG равным 0. Это позволяет логике загрузить соответствующее значение из первого контроллера в первый элемент массива *local_array*.

Соответствие IEC61131-3

Использование этого приложения

Введение

Международная электротехническая комиссия (IEC) разработала ряд спецификаций для программируемых контроллеров. Эти спецификации должны содействовать международной унификации оборудования и языков программирования, предназначенных для использования в индустрии средств управления. Эти стандарты служат в качестве основы для контроллеров Logix5000 и пакета программирования RSLogix 5000.

Спецификация IEC для программируемого контроллера разделена на пять отдельных частей, каждая из которых посвящена определенному аспекту системы управления.

- Часть 1: Общая информация
- Часть 2: Оборудование и проверка на соответствие техническим требованиям
- Часть 3: Языки программирования
- Часть 4: Руководство пользователя
- Часть 5: Спецификация на сервис обмена сообщениями

Для индустрии средств управления в целом наибольший интерес представляет часть 3 (IEC61131-3) «Языки программирования», поскольку именно эта часть является краеугольным камнем для выполнения других стандартов и обеспечивает наиболее весомую выгоду для конечного пользователя посредством снижения затрат на обучение. В связи с этим здесь рассматривается лишь IEC61131-3.

Спецификация IEC61131-3 по языкам программирования рассматривает различные аспекты программируемого контроллера, включая работу операционной системы, определение данных, языки программирования и набор инструкций. Компоненты спецификации IEC61131-3 отнесены к различным категориям в зависимости от того, являются ли они обязательным требованием, не обязательными или дополнительными. Таким образом спецификация IEC61131-3 обеспечивает минимальный набор функциональных возможностей, который может быть расширен для удовлетворения нужд приложения конечного пользователя. Другой стороной такого подхода является то, что каждый производитель программируемых систем управления может реализовывать различные компоненты спецификации или предоставлять различные дополнительные возможности.

Операционная система

Многозадачная операционная система с приоритетным прерыванием (ОС) контроллеров Logix5000 соответствует определению IEC61131-3. В IEC61131-3 указывается, что ОС программируемых контроллеров может содержать ноль или более задач, которые могут выполнять одну или более программ. Каждая из программ может содержать одну или более функций или процедур. Согласно IEC61131-3, количество каждого из этих компонентов зависит от реализации. Контроллеры Logix5000 обеспечивают выполнение нескольких задач, каждая из которых содержит несколько программ и неограниченное количество функций и процедур.

IEC61131-3 обеспечивает возможность создания различных классов задач по их выполнению. Задачи могут быть сконфигурированы как непрерывные, периодические или событийные. Непрерывную задачу не нужно планировать, она будет использовать любое оставшееся время обработки, когда другие задачи неактивны. Работа периодических задач планируется на основе повторяющегося временного интервала. Спецификация IEC61131-3 не задает базовое время для конфигурирования периодической задачи. Событийная задача IEC61131-3 запускается при обнаружении переднего фронта сконфигурированного входа. Контроллеры Logix5000 поддерживают выполнение и непрерывных, и периодических задач. Кроме того, период для периодической задачи можно конфигурировать начиная с минимального значения в одну миллисекунду (мс).

Определение данных

Спецификация IEC61131-3 обеспечивает доступ к памяти посредством создания именованных переменных. Согласно IEC61131-3, имена переменных состоят минимум из шести символов (программное обеспечение RSLogix5000 поддерживает минимум один символ), начиная с символа подчеркивания «_» или текстового символа (A-Z), за которым следуют один или более символов, включая символ подчеркивания «_», текстовый символ (A-Z) или цифру (0-9). По желанию, если имена не чувствительны к регистру (A = a, B = b, C = c ...), могут также подерживаться текстовые символы нижнего регистра (a-z). Контроллеры Logix5000 обеспечивают полное соответствие этому определению, поддерживают опцию нижнего регистра и позволяют увеличивать размер имени до 40 символов.

Переменные в IEC61131-3 могут задаваться таким образом, чтобы они были доступны всем программам внутри ресурса или контроллера, или чтобы обеспечивался ограниченный доступ лишь к функциям или процедурам внутри одной программы. Чтобы передавать данные между несколькими ресурсами или контроллерами, могут быть сконфигурированы пути доступа для определения местоположения данных внутри системы. Контроллеры Logix5000 обеспечивают соответствие этому требованию посредством данных в области видимости программы и данных в области видимости контроллера и позволяют конфигурировать пути доступа при помощи производимых/потребляемых данных.

Интерпретация переменной в памяти в IEC61131-3 определяется посредством использования либо элементарного типа данных, либо дополнительного производного типа данных, создаваемого на основе группы нескольких типов данных. Контроллеры Logix5000 поддерживают использование таких элементарных типов данных, как BOOL (1бит), SINT (8-битовое целое число), INT (16-битовое целое число), DINT (32-битовое целое число) и REAL (число с плавающей точкой IEEE). Кроме того, дополнительные производные типы данных поддерживаются через создание задаваемых пользователем структур и массивов.

Языки программирования

Спецификация IEC61131-3 определяет пять (5) различных языков программирования и набор общих элементов. Все языки определены как не обязательные к использованию, но как минимум один язык должен поддерживаться, чтобы утверждать о соответствии с данной спецификацией. Компоненты языка программирования в IEC61131-3 определены следующим образом:

- Общие элементы языка
- Общие графические элементы
- Элементы языка перечня инструкций
- Элементы языка структурированного текста
- Элементы языка релейной логики
- Элементы языка последовательной функциональной схемы
- Элементы языка функциональной блок-схемы

Контроллеры Logix5000 и программное обеспечение RSLogix5000 обеспечивают поддержку общих элементов языка, а также опций языка структурированного текста, релейной логики, последовательной функциональной схемы и функциональной блок-схемы. Кроме того, в данной среде используется формат импорта/экспорта ASCII, основанный на языке структурированного текста. Набор инструкций и функции обмена программными файлами подробно рассматриваются в следующих разделах настоящего руководства.

Набор инструкций

Задаваемый IEC61131-3 набор инструкций является полностью произвольным. В спецификации представлен ограниченный набор инструкций, при реализации которого должно быть обеспечено соответствие указанному выполнению и визуальному представлению. Однако IEC61131-3 не ограничивает набор инструкций только теми инструкциями, которые указаны в спецификации. Каждый производитель программируемых контроллеров может реализовать дополнительные функции в виде инструкций помимо тех инструкций, которые указаны в спецификации. Примерами таких расширенных инструкций являются инструкции, необходимые для выполнения диагностики, пропорционально-интегрально-дифференциального управления циклом, управления перемещениями, а также манипуляций с файлами данных. Поскольку расширенные инструкции не определены спецификацией IEC61131-3, нет гарантии совместимости реализаций между различными производителями программируемых контроллеров. Поэтому использование таких инструкций может препятствовать перемещению логики между производителями.

Контроллеры Logix5000 и программное обеспечение RSLogix5000 предоставляют комплект инструкций, выполняющихся так, как определено спецификацией IEC61131-3. Физическое представление этих инструкций соответствует пользовательскому интерфейсу существующих систем, что позволяет уменьшить расходы на обучение работе в этой среде. Кроме инструкций, соответствующих IEC61131-3, в эту среду перенесен целый ряд инструкций из существующих продуктов, с тем чтобы не была потеряна ни одна функциональная возможность.

Мобильность программ IEC61131-3

Одной из задач конечных пользователей, создающих программы в среде, соответствующей IEC61131-3, является обеспечение переносимости, или мобильности, программ между контроллерами, создаваемыми различными производителями. В этой области IEC61131-3 имеет недостатки, поскольку эта спецификация не определяет ни одного формата обмена файлами. Это означает, что любая программа, созданная в среде одного производителя, требует манипуляций для перемещения в систему другого производителя.

Чтобы минимизировать усилия по переносу программ между оборудованием различных производителей, пакет программирования контроллеров RSLogix 5000 включает полную утилиту экспорта и импорта ASCII. Кроме того, файловый формат, используемый этим инструментом, представляет собой гибрид определения языка структурированного текста IEC61131-3. Операционная система и определения данных контроллера соответствуют форматам IEC61131-3. Расширения реализованы для преобразования релейной логики в текст ASCII, поскольку это не определено IEC61131-3.

Для получения дополнительной информации об утилите экспорта и импорта ASCII пакета RSLogix 5000 обратитесь к справочному руководству *Logix5000 Controllers Import/Export Reference Manual* (Справочное руководство по экспорту/импорту для контроллеров Logix5000), публикация 1756-RM084.

Таблицы соответствия IEC

Контроллеры Logix5000 и программное обеспечение RSLogix5000 соответствует требованиям IEC61131-3 по следующим функциям языка:

Номер таблицы:(1)	Номер функции:	Описание функции:	Примечания по расширениям и реализации:
1	2	Буквы нижнего регистра	нет
1	3a	Знак номера (#)	Используется для обозначения типа данных непосредственных значений
1	4a	Знак доллара ($)	Используется для символа описания и управления строкой
1	6a	Ограничители нижнего индекса ([])	Индексы массива
2	1	Идентификаторы, использующие верхний регистр и цифры.	Имена задач, программ, процедур, структур и тегов
2	2	Идентификаторы, использующие верхний регистр, цифры и встроенное подчеркивание	Имена задач, программ, процедур, структур и тегов
2	3	Идентификаторы, использующие верхний и нижний регистры, цифры и встроенное подчеркивание	Имена задач, программ, процедур, структур и тегов
3	1	Комментарии	Комментарий структурированного текста (CT), также поддерживается / *комментарий*/ и комментарий в конце строки после //.
4	1	Целочисленный литерал	12, 0, -12
4	2	Действительный литерал	12.5, -12.5
4	3	Действительный литерал с экспонентами	-1.34E-12, 1.234E6
4	4	Литерал с основанием 2	2#0101_0101
4	5	Литерал с основанием 8	8#377
4	6	Литерал с основанием 16	16#FFE0
4	7	Булевы ноль и единица	0, 1
5	1A	Пустая строка ”	Описания и редактор строки
5	1B	Строка длины 1, содержащая символ ‘A’	Описания и редактор строки
5	1C	Строка длины 1, содержащая пробел ‘ ‘	Описания и редактор строки
5	1D	Строка длины 1, содержащая символ одинарной кавычки ‘$”	Описания и редактор строки
5	1E	Строка длины 1, содержащая символ двойных кавычек ‘”’	Описания и редактор строки
5	1F	Строка длины 2, содержащая символы CR и LF	Описания и редактор строки
5	1G	Строка длины 1, содержащая символ LF ‘$0A’	Описания и редактор строки
5	1H	Строка длины 5, которая будет выводиться на печать как “$1.00”, используя ‘$$1.00’	Описания и редактор строки
5	1I	Эквивалентные строки длины 2 ‘AE’ и ‘CCB’	Описания и редактор строки
6	2	Строковый знак доллара ‘$$’	Описания и редактор строки
6	3	Строковая одинарная кавычка ‘$”	Описания и редактор строки
6	4	Перевод строки ‘$L’ или ‘$l’	Описания и редактор строки

Номер таблицы:(1)	Номер функции:	Описание функции:	Примечания по расширениям и реализации:
6	5	Новая строка ‘$N’ или ‘$n’	Описания и редактор строки
6	6	Новая страница ‘$P’ или ‘$p’	Описания и редактор строки
6	7	Возврат каретки для строки ‘$R’ или ‘$r’	Описания и редактор строки
6	8	Табуляция строки ‘$T’ или ‘$t’	Описания и редактор строки
6	9	Двойные кавычки строки $”	Описания и редактор строки
10	1	Тип данных BOOL	Определение теговой переменной
10	2	Тип данных SINT	Определение теговой переменной
10	3	Тип данных INT	Определение теговой переменной
10	4	Тип данных DINT	Определение теговой переменной
10	10	Тип данных REAL	Определение теговой переменной
10	12	Время	Определение теговой переменной, структура TIMER
10	16	Тип данных STRING	8 бит
11	1	Иерархия типов данных	нет
12	1	Непосредственные производные от элементарных типов	Структуры пользовательских типов данных
12	4	Типы данных массива	Определение теговой переменной
12	5	Типы структурированных данных	Структуры пользовательских типов данных
13	1	Исходное значение 0 для BOOL, SINT, INT, DINT	Определение теговой переменной
13	4	Исходное значение 0.0 для REAL, LREAL	Определение теговой переменной
13	5	Исходное значение времени T#0s	Структуры пользовательских типов данных
13	9	Пустая строка ”	Описания и строки
14	1	Инициализация непосредственно производных типов	Импорт/экспорт
14	4	Инициализация типов данных массива	Импорт/экспорт
14	5	Инициализация элементов структурированного типа	Импорт/экспорт
14	6	Инициализация производных типов структурированных данных	Импорт/экспорт
19a	2a	Текстовый вызов, неформальный	Имеется в СТ
20	1	Использование с EN и ENO	Функция имеется в релейной логике (РЛ), но не помечена. Имеется в функциональной блок-схеме (ФБС).
20	2	Использование без EN и ENO	Имеется в ФБС
20	3	Использование с EN, но без ENO	Имеется в ФБС
20	4	Использование без EN, но с ENO	Имеется в ФБС
21	1	Переопределяемые функции ADD(INT, DINT) или ADD(DINT, REAL)	Описание всех поддерживаемых переопределяемых типов сопровождает каждую инструкцию
22	1	Функция преобразования _TO_	Инструкции RAD, DEG. Преобразование радиан в десятичные числа и обратно. Численное преобразование строк STOR, RTOS, DTOS. Другие не требуются ввиду переопределения инструкций
22	2	Функция преобразования путем усечения	Инструкция TRN в РЛ и TRUNC в СТ
22	3	Преобразование BCD (двоично-десятичных чисел) в INT	Инструкция FRD в РЛ

Номер таблицы:[1]	Номер функции:	Описание функции:	Примечания по расширениям и реализации:
22	4	Преобразование INT в BCD	Инструкция TOD в РЛ
23	1	Абсолютная величина	Инструкция ABS
23	2	Квадратный корень	Инструкция SQR в РЛ и ФБС, и функция SQRT в СТ
23	3	Натуральный логарифм	Инструкция LN
23	4	Десятичный логарифм	Инструкция LOG
23	6	Синус в радианах	Инструкция/функция SIN
23	7	Косинус в радианах	Инструкция/функция COS
23	8	Тангенс в радианах	Инструкция/функция TAN
23	9	Основной арксинус	Инструкция ASN в РЛ и ФБС, и функция ASIN в СТ
23	10	Основной арккосинус	Инструкция ACS в РЛ и ФБС, и функция ACOS в СТ
23	11	Основной арктангенс	Инструкция ATN в РЛ и ФБС, и функция ATAN в СТ
24	12	Арифметическое сложение	Инструкция ADD в РЛ и ФБС, и + в СТ
24	13	Арифметическое умножение	Инструкция MUL в РЛ и ФБС, и * в СТ
24	14	Арифметическое вычитание	Инструкция SUB в РЛ и ФБС, и - в СТ
24	15	Арифметическое деление	Инструкция DIV в РЛ и ФБС, и / в СТ
24	16	Остаток от деления	Инструкция MOD в РЛ и ФБС
24	17	Возведение в степень	Инструкция XPY в РЛ и ФБС, и ** в СТ
24	18	Перемещение значения	Инструкция MOV в РЛ, и := в СТ
25	1	Сдвиг бита влево	Функция, содержащаяся в инструкции BSL в РЛ для сдвига 1
25	2	Сдвиг бита вправо	Функция, содержащаяся в инструкции BSR в РЛ для сдвига 1
25	3	Циклический сдвиг бита влево	Функция, содержащаяся в инструкции BSL в РЛ для сдвига 1
25	4	Циклический сдвиг бита вправо	Функция, содержащаяся в инструкции BSR в РЛ для сдвига 1
26	5	AND	Инструкция BAND в ФБС, и оператор «&» в СТ
26	6	OR	Инструкция BOR в ФБС
26	7	XOR	Инструкция BXOR в ФБС
26	8	NOT	Инструкция BNOT в ФБС
27	1	SELECT	Инструкция SEL в ФБС
27	2a	Выбор максимального значения MAX	Функция, содержащаяся в инструкции ASEL в ФБС и СТ
27	2b	Выбор минимального значения MIN	Функция, содержащаяся в инструкции ASEL в ФБС и СТ

Номер таблицы:[1]	Номер функции:	Описание функции:	Примечания по расширениям и реализации:
27	3	Высокий/низкий предел LIMIT	Инструкция HLL в ФБС и CT
27	4	Мультиплексор MUX	Инструкция MUX в ФБС
28	5	Сравнение: больше	Инструкция GRT в РЛ и ФБС, и > в CT
28	6	Сравнение: больше или равно	Инструкция GRE в РЛ и ФБС, и >= в CT
28	7	Сравнение: равно	Инструкция EQU в РЛ и ФБС, и = в CT
28	8	Сравнение: меньше	Инструкция LES в РЛ и ФБС, и < в CT
28	9	Сравнение: меньше или равно	Инструкция LEQ в РЛ и ФБС, и <= в CT
28	10	Сравнение: неравно	Инструкция NEQ в РЛ и ФБС, и <> в CT
29	1	Длина строки LEN	Содержится как параметр в типе данных STRING
29	4	Средняя строка MID	Инструкция MID в РЛ и CT
29	5	Конкатенация строк CONCAT	Инструкция CONCAT в РЛ и CT
29	6	Вставка строки INSERT	Инструкция INSERT в РЛ и CT
29	7	Удаление строки DELETE	Инструкция DELETE в РЛ и CT
29	9	Поиск строки FIND	Инструкция FIND в РЛ и CT
32	1	Чтение входа	ФБС и CT
32	2	Запись входа	ФБС и CT
32	3	Чтение выхода	ФБС и CT
32	4	Запись выхода	ФБС и CT
34	1	Бистабильная доминанта установки	Инструкция SETD в ФБС и CT
34	2	Бистабильная доминанта переустановки	Инструкция RESD в ФБС и CT
35	1	Детектор переднего фронта	Инструкция OSR в РЛ и инструкция OSRI в ФБС и CT
35	2	Детектор заднего фронта	Инструкция OSF в РЛ и инструкция OSFI в ФБС и CT
36	1b	Счетчик прямого счета	Функция, содержащаяся в инструкциях CTU и RES в РЛ и в инструкции CTUD в ФБС и CT
37	2a	Таймер по включении	Функция, содержащаяся в инструкции TON в РЛ и в инструкции TONR в ФБС и CT
37	3a	Таймер по выключении	Функция, содержащаяся в инструкции TOF в РЛ и в инструкции TOFR в ФБС и CT
38	2	Отсчет времени по включении	Функция, содержащаяся в инструкции TON в РЛ и в инструкции TONR в ФБС и CT
38	3	Отсчет времени по выключении	Функция, содержащаяся в инструкции TOF в РЛ и в инструкции TOFR в ФБС и CT
40	1a	Шаг последовательной функциональной схемы (ПФС)	
40	1b	Исходный шаг ПФС	
40	2a	Шаг ПФС, текстовый	Импорт/экспорт, имя шага задается при помощи формата “Operand := step_name” (“Операнд := имя_шага”)

Номер таблицы:[1]	Номер функции:	Описание функции:	Примечания по расширениям и реализации:
40	2b	Исходный шаг ПФС, текстовый	Импорт/экспорт, использует параметр “InitialStep” (исходный шаг), имя шага задается с помощью формата “Operand := step_name”
40	3a	Общая форма флага шага ПФС	Тег, поддерживающий шаг
40	4	Общая форма времени выполнения шага	Тег, поддерживающий шаг
41	1	Переход посредством СТ	
41	5	Текстовая форма перехода	Импорт/экспорт с различным форматированием
41	7	Имя перехода	Тег, поддерживающий переход
41	7a	Переход, задаваемый РЛ	Тег, поддерживающий переход
41	7b	Переход, задаваемый ФБС	Тег, поддерживающий переход
41	7d	Переход, задаваемый СТ	Тег, поддерживающий переход
42	1	Булево действие	Тег, поддерживающий действие
42	3s	Текстовое представление действия	Импорт/экспорт
43	1	Привязка действия шага	
43	2	Шаг со сцепленными действиями	
43	3	Тело текстового шага	Импорт/экспорт с различным форматированием
43	4	Поле тела действия	Встроенный СТ
44	1	Определитель блока действий	
44	2	Имя блока действий	
44	3	Тег индикатора действия	Расширено для поддержки DINT, INT, SINT и REAL дополнительно к BOOL
44	5	Действие, использующее СТ	Поддерживает как встроенный СТ, так и процедуру JSR для перехода к СТ
44	6	Действие, использующее РЛ	С использованием процедуры JSR для перехода РЛ
44	7	Действие, использующее ФБС	С использованием процедуры JSR для перехода к ФБС
45	1	Отсутствие определителя действия	По умолчанию установлен на N, когда в явном виде ничего не введено
45	2	Определитель действия N – отсутствие сохранения	
45	3	Определитель действия R – сброс	
45	4	Определитель действия S – установлен/ сохранен	
45	5	Определитель действия L – ограничение времени	
45	6	Определитель действия D – задержка времени	
45	7	Определитель действия P – импульс	
45	8	Определитель действия SD – сохранение и задержка времени	
45	9	Определитель действия DS – задержка и сохранение	
45	10	Определитель действия SL – сохранение и ограничение времени	
45	11	Определитель действия P1 – передний фронт импульса	
45	12	Определитель действия P0 – задний фронт импульса	

Номер таблицы:(1)	Номер функции:	Описание функции:	Примечания по расширениям и реализации:
45a	1	Управление действием	
45a	2	Управление действием	
46	1	Одна последовательность ПФС	
46	2a	Расходимость ПФС при выборе последовательности	Использование соединений линий вместо звездочки
46	2b	Расходимость ПФС при выборе последовательности с порядком выполнения	
46	3	Сходимость ПФС при выборе последовательности	
46	4a	Одновременная расходимость последовательностей ПФС	
46	4b	Одновременная сходимость последовательностей ПФС	
46	5a, b, c	Пропуск последовательности ПФС	
46	6a, b, c	Цикл последовательностей ПФС	
46	7	Стрелки направления цикла ПФС	При невидимых проводных соединениях
47	1	Графическое представление ПФС	
47	4	Графическое представление ПФС	
48	1	Минимальные требования соответствия шагов ПФС	Обратитесь к примечаниям в отдельных таблицах выше.
48	2	Минимальные требования соответствия перехода ПФС	Обратитесь к примечаниям в отдельных таблицах выше.
48	3	Минимальные требования соответствия действий ПФС	Обратитесь к примечаниям в отдельных таблицах выше.
48	4	Минимальные требования соответствия тела действия ПФС	Обратитесь к примечаниям в отдельных таблицах выше.
48	5	Минимальные требования соответствия описателя действия ПФС	Обратитесь к примечаниям в отдельных таблицах выше.
48	6	Минимальные требования соответствия ветвления ПФС	Обратитесь к примечаниям в отдельных таблицах выше.
48	7	Минимальные требования соответствия соединения блоков ПФС	Обратитесь к примечаниям в отдельных таблицах выше.
55	1	Заключение CT в круглые скобки (выражение)	
55	2	Оценка функции CT	Использование неформальной формы вызова для встроенных функций. JSR используется внутри языка CT для вызова разработанного пользователем кода.
55	3	Возведение в степень CT **	
55	4	Отрицание CT –	
55	5	Отрицание CT NOT	
55	6	Умножение CT *	
55	7	Деление CT /	
55	8	Остаток от деления CT MOD	
55	9	Сложение CT +	
55	10	Вычитание CT -	
55	11	Сравнение CT <, >, <=, >=	

Номер таблицы:(1)	Номер функции:	Описание функции:	Примечания по расширениям и реализации:
55	12	Равенство CT =	
55	13	Неравенство CT <>	
55	14	Булево И CT в виде &	
55	15	Булево И	
55	16	Булево исключающее ИЛИ CT	
55	17	Булево ИЛИ	
56	1	Булево присвоение CT :=	
56	2	Вызов функционального блока CT	
56	3	CT RETURN	RET( ) с несколькими параметрами
56	4	CT IF / ELSIF / ELSE / END_IF	
56	5	CT CASE OF /ELSE / END_CASE	
56	6	CT FOR / END_FOR	
56	7	CT WHILE DO / END_WHILE	
56	8	CT REPEATE / UNTIL / END_REPEATE	
56	9	CT EXIT	
56	10	Пустой оператор CT ;	
57	1, 2	Горизонтальная линия	Редактор РЛ, редактор ФБС
57	3, 4	Вертикальная линия	Редактор РЛ, редактор ФБС
57	5, 6	Горизонтальное/вертикальное соединение	Редактор РЛ, редактор ФБС
57	7, 8	Пересечения линий без соединения	Редактор ФБС
57	9, 10	Соединительные и не соединительные углы	Редактор РЛ, редактор ФБС
57	11, 12	Блоки с соединениями	Редактор РЛ, редактор ФБС
57	13, 14	Коннекторы	Редактор ФБС
58	2	Безусловный переход	Инструкция JMP в РЛ
58	3	Цель перехода	Инструкция LBL в РЛ
58	4	Условный переход	Инструкция JMP в РЛ
58	5	Условный возврат	Инструкция RET в РЛ
58	8	Безусловный возврат	Инструкция RET в РЛ
59	1	Левая шина питания	Редактор РЛ
59	2	Правая шина питания	Редактор РЛ
60	1	Горизонтальная связь	Редактор РЛ
60	2	Вертикальная связь	Редактор РЛ
61	1, 2	Нормально разомкнутый контакт --| |--	Инструкция XIC в РЛ
61	3, 4	Нормально замкнутый контакт --|/|--	Инструкция XIO в РЛ
61	5, 6	Чувствительный контакт положительного перехода -|P|-	Инструкция ONS в РЛ
62	1	Обмотка --( )--	Инструкция OTE в РЛ

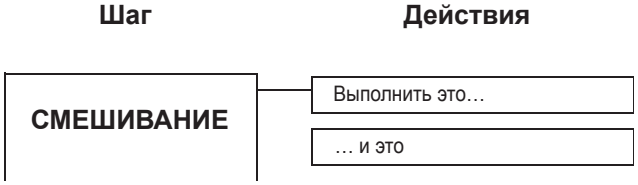
Номер таблицы:(1)	Номер функции:	Описание функции:	Расширения и примечания реализации:
62	3	Установка (фиксация) обмотки	Функция, содержащаяся в инструкции OTL в РЛ
62	4	Сброс (снятие фиксации) обмотки	Функция, содержащаяся в инструкции OUT в РЛ
62	8	Чувствительная обмотка положительного перехода	Инструкция OSR в РЛ
62	9	Чувствительная обмотка отрицательного перехода	Инструкция OSF в РЛ

(1) Таблицы, относящиеся к языкам, отличным от структурированного текста, последовательной функциональной схемы, релейной логики и функциональной блок-схемы, пропущены.

A

action (действие)

В последовательной функциональной схеме (ПФС) действие представляет собой функциональное разбиение шага. Несколько действий составляют один шаг. Каждое действие выполняет конкретную функцию, например, управление мотором, открытие клапана или перевод группы устройств в определенный режим.



Каждое действие включает в себя определитель. Когда шаг активен (выполняется), то определитель задает, когда начинается и прекращается действие.

См. *sequential function chart* (последовательная функциональная схема), *step* (шаг), *qualifier* (определитель).

alias tag (тег-псевдоним)

Тег, ссылающийся на другой тег. Тег-псевдоним может ссылаться на другой тег-псевдоним или на базовый тег. Тег-псевдоним также может ссылаться на компонент другого тега путем ссылки на член структуры, элемент массива или бит внутри тега или члена. См. *base tag* (базовый тег).

ASCII

7-ми разрядный код (с дополнительным разрядом четности), который используется для представления буквенно-цифровых символов, знаков пунктуации, а также символов управляющих кодов. Перечень кодов ASCII находится на задней стороне обложки этого руководства.

asynchronous (асинхронный)

Действия, которые происходят независимо друг от друга и не имеют регулярный характер. В контроллерах Logix5000 значения ввода/вывода обновляются асинхронно по отношению к выполнению логики:

- Программы внутри задачи обращаются к входным и выходным данным непосредственно из памяти в области видимости контроллера.
- Логика внутри любой задачи может модифицировать данные в области видимости контроллера.
- Данные и значения ввода/вывода асинхронны и могут изменяться в процессе выполнения задачи.
- Входное значение, на которое делается ссылка в начале выполнения задачи, может быть другим при последующей ссылке на него.

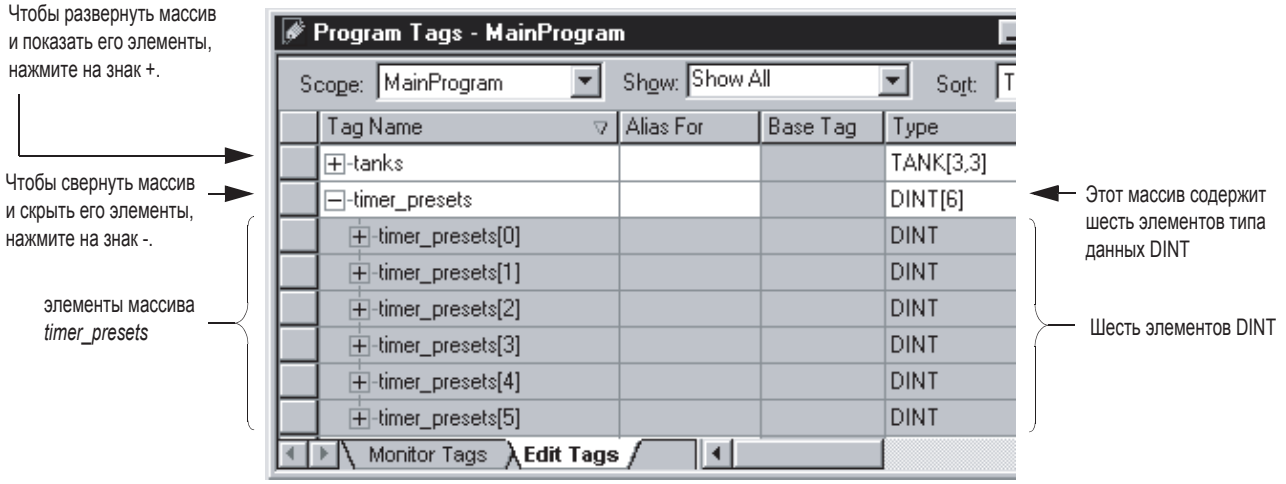
ВНИМАНИЕ

Позаботьтесь о том, чтобы память данных содержала соответствующие значения в течение всего времени выполнения задачи. Вы можете сделать копию данных или поместить их буфер в начале сканирования, чтобы обеспечить опорные значения для вашей логики.

array (массив)

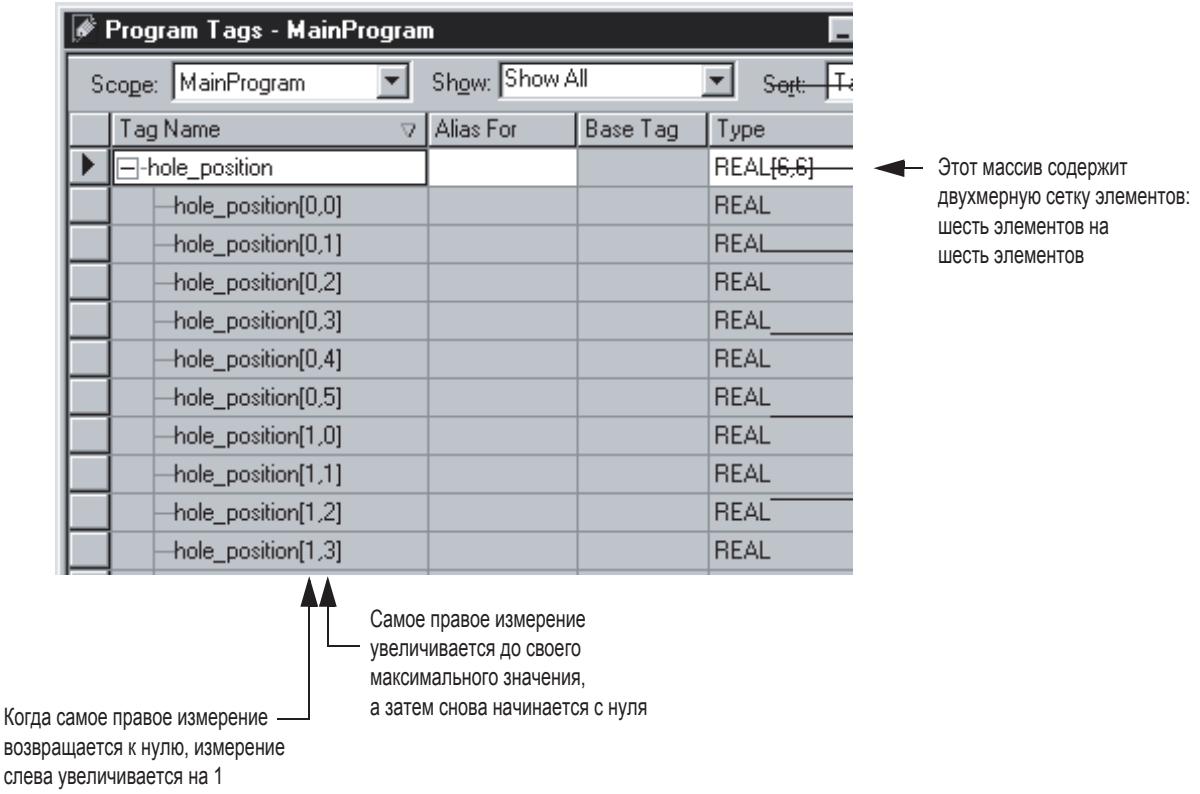
Массив позволяет вам группировать данные (одного типа данных) под общим именем.

- Массив аналогичен файлу.
- Нижний индекс (s) идентифицирует каждый отдельный **элемент** внутри массива.
- Наименьшее значение нижнего индекса – 0, наибольшее значение соответствует количеству элементов минус 1 (так как отсчет начинается с 0).

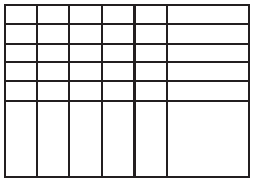
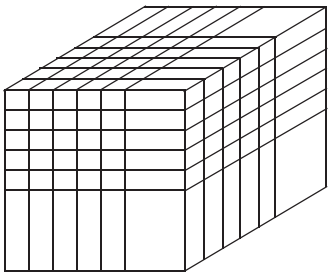


- Тег массива занимает непрерывный блок памяти в контроллере, при этом элементы идут последовательно один за другим.
- Вы можете использовать инструкции массива и секвенсера для манипулирования с элементами массива или их индексации.
- Массив может иметь три измерения. Это дает вам гибкость при обозначении элемента при помощи одного, двух или трех нижних индексов (координат).

- В двумерном или трехмерном массиве самое правое измерение первым увеличивается в памяти.



- Общее количество элементов в массиве является произведением размеров по всем измерениям, как показано в следующих примерах:

Этот массив:	Сохраняет данные следующим образом:	Например:				
одномерный		Имя тега:	Тип	Измерение 0	Измерение 1	Измерение 2
		one_d_array	DINT[7]	7	--	--
		общее количество элементов = 7 допустимый диапазон нижнего индекса DINT[x], где x=0-6				
двумерный		Имя тега:	Тип	Измерение 0	Измерение 1	Измерение 2
		two_d_array	DINT[4,5]	4	5	--
		общее количество элементов = 4 * 5 = 20 допустимый диапазон нижнего индекса DINT[x,y], где x=0-3; y=0-4				
трехмерный		Имя тега:	Тип	Измерение 0	Измерение 1	Измерение 2
		three_d_array	DINT[2,3,4]	2	3	4
		общее количество элементов = 2 * 3 * 4 = 24 допустимый диапазон нижнего индекса DINT[x,y,z], где x=0-1; y=0-2, z=0-3				

- Вы можете изменять измерения массива в процессе программирования в режиме оффлайн без потери данных тега. Вы не можете изменять измерения массива, работая в режиме онлайн.

application (приложение)

Совокупность процедур, программ, задач, а также конфигурация ввода/вывода, используемые для задания работы одного контроллера. См. *project* (проект).

B

base tag (базовый тег)

Тег, который фактически определяет память, где хранится элемент данных. См. *alias tag* (тег-псевдоним).

bidirectional connection (двунаправленное соединение)

Соединение, в котором данные перемещаются в обоих направлениях: от источника к получателю и от получателя к источнику. См. *connection* (соединение), *unidirectional connection* (однонаправленное соединение).

binary (двоичный)

Целочисленные значения, отображаемые и вводимые в двоичной системе счисления (каждый разряд представляет собой один бит). Имеют префикс 2#. Заполняют длину булева или целочисленного значения (1, 8, 16 или 32 бита). При отображении каждая группа из четырех разрядов отделяется знаком подчеркивания для большей удобочитаемости. См. *decimal* (десятичный), *hexadecimal* (шестнадцатеричный), *octal* (восьмеричный).

bit (бит)

Двоичный разряд. Самая маленькая единица памяти. Отображается цифрами 0 (сброшен) и 1 (установлен).

BOOL

Тип данных, который сохраняет состояние одного бита, где:

- *0* соответствует *off* (выключено)
- *1* соответствует *on* (включено)

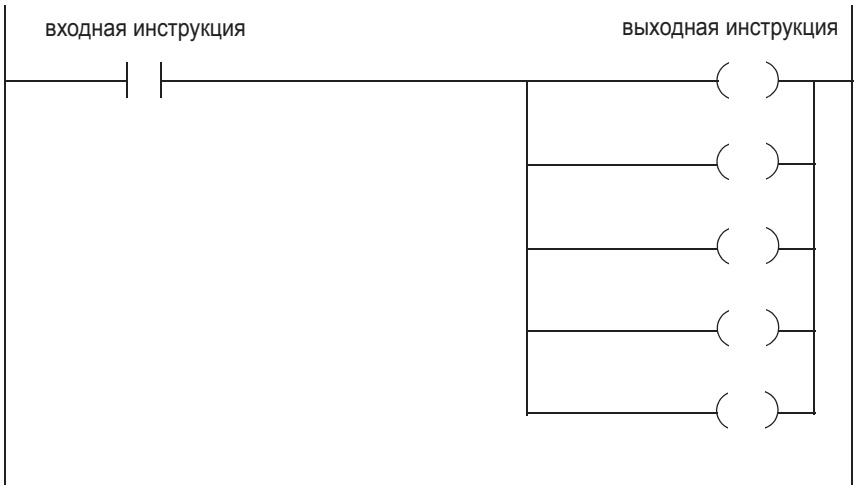
BOOL expression (выражение BOOL)

Выражение в структурированном тексте, дающее значение BOOL, равное 1 (истина) или 0 (ложь).

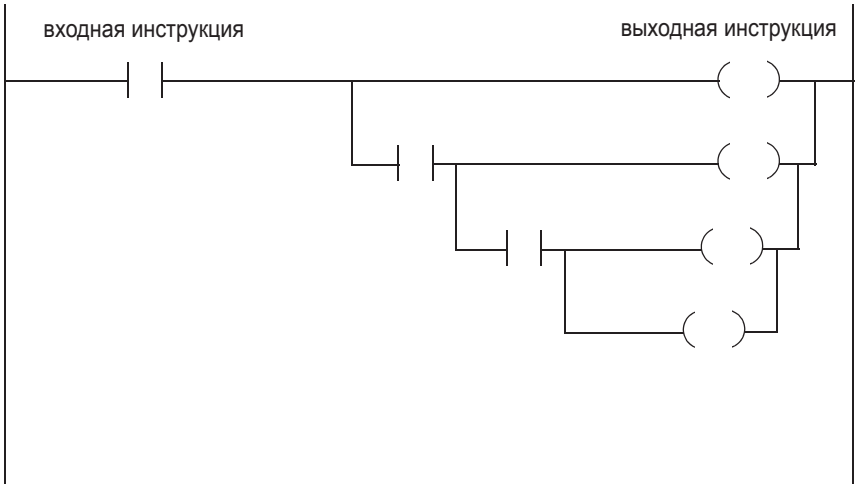
- Булево выражение использует булевы теги, операторы отношения и логические операторы для сравнения значений или проверки условий на ложь или истину. Например, tag1 > 65.
- Простое булево выражение может представлять собой один тег BOOL.
- Обычно булевы выражения используются для задания условий выполнения другой логики.

branch (ветвь)

Не существует ограничений по количеству уровней параллельных ветвей, которые вы можете ввести. На следующем рисунке показана параллельная ветвь с пятью уровнями. Главная цепочка – это первый уровень ветвления, за ним следуют четыре дополнительные ветви.



Вы можете вкладывать ветви вплоть до шести уровней. На следующем рисунке показана вложенная ветвь. Нижняя выходная инструкция находится на вложенной ветви глубиной в три уровня.



byte (байт)

Единица памяти, состоящая из 8 бит.

C

cache (кэш)

В зависимости от того, как вы сконфигурировали инструкцию MSG, эта инструкция может использовать соединение для отправки и получения данных.

Этот тип сообщения:	И этот метод связи:	Используют соединение:
считывание или запись таблицы данных CIP	→	✔
PLC2, PLC3, PLC5 или SLC (все типы)	CIP CIP с Source ID	
	DH+	✔
CIP generic (общий)	→	ваш выбор(1)
считывание или запись с поблочной передачей	→	✔

(1) Вы можете установить соединение для сообщений CIP generic. Но для большинства приложений мы рекомендуем вам оставить сообщения CIP generic без соединения.

Если инструкция MSG использует соединение, вы можете оставить это соединение открытым (кэшировать) или закрыть его, когда сообщение будет передано.

Если вы:	То:
Кэшируете соединение	Соединение остается открытым после завершения выполнения инструкции MSG. Это оптимизирует время выполнения. Открытие соединения при каждом выполнении сообщения увеличивает время выполнения.
Не кэшируете соединение	Соединение закрывается после завершения выполнения инструкции MSG. Это освобождает соединение для использования в других целях.

Контроллер имеет следующие пределы по количеству соединений, которые вы можете кэшировать:

Если у вас эта ревизия программного и микропрограммного обеспечения:	То вы можете кэшировать:
11.x или более ранняя	• сообщения поблочной передачи для максимум 16 соединений • другие типы сообщений для максимум 16 соединений
12.x или более поздняя	до 32 соединений

Если несколько сообщений отправляются в одно и то же устройство, возможно, эти сообщения смогут совместно использовать одно соединение.

Если инструкции MSG адресованы:	И они:	То:
различным устройствам	→	Каждая инструкция MSG использует 1 соединение.
одному устройству	разрешены одновременно	Каждая инструкция MSG использует 1 соединение.
	НЕ разрешены одновременно	Инструкции MSG совместно используют 1 соединение (то есть, все инструкции MSG вместе считаются одним соединением).

ПРИМЕР

Совместное использование соединения

Если контроллер отправляет в один и тот же модуль то сообщение считывания с поблочной передачей, то сообщение записи с поблочной передачей, то вместе эти сообщения считаются одним соединением. Кэширование обоих сообщений считается за одно в списке кэширований.

См. *connection* (соединение), *uncached connection* (не кэшированное соединение).

change of state (COS) (изменение состояния)

Любое изменение состояния точки или группы точек в модуле ввода/вывода.

CIP

См. Control and Information Protocol (протокол управления и информации).

communication format (формат обмена данными)

Задает, каким образом модуль ввода/вывода обменивается данными с контроллером. Выбор формата обмена данными определяет:

- какие закладки конфигурирования доступны через пакет программирования
- структуру тега и метод конфигурирования

compatible module (совместимый модуль)

Режим защиты посредством электронного ключа, который требует, чтобы производитель, номер по каталогу и атрибуты основной ревизии физического модуля и модуля, сконфигурированного в программном обеспечении, совпадали, чтобы установить соединение с модулем. См. *disable keying* (отмена ключа), *exact match* (точное соответствие).

connection (соединение)

Линия связи между двумя устройствами, например, между контроллером и модулем ввода/вывода, терминалом PanelView или другим контроллером.

- Соединения – это распределение ресурсов, обеспечивающее более надежную связь между устройствами, чем сообщения без соединения.
- Количество соединений, которые может иметь один контроллер, ограничено.
- Вы косвенно задаете количество используемых контроллером соединений, когда конфигурируете контроллер для обмена данными с другими устройствами в системе.

consumed tag (потребляемый тег)

Тег, который получает данные, рассылаемые производимым тегом по сети ControlNet или через объединительную плату ControlLogix. Потребляемый тег должен быть:

- в области видимости контроллера
- того же типа данных (включая измерения массива), что и удаленный тег (производимый тег)

См. *produced tag* (производимый тег).

continuous task (непрерывная задача)

Задача, которая выполняется непрерывно

- Непрерывная задача выполняется в фоновом режиме. Всякое время центрального процессора, не занятое для других операций, (таких как перемещение, обмен данными и периодические задачи) используется для выполнения программ непрерывной задачи.
- Непрерывная задача перезапускается после того, как последняя из ее программ закончит выполнение.
- Для проекта непрерывная задача не является обязательной.
- Может использоваться только одна непрерывная задача.
- Все периодические задачи прерывают выполнение непрерывной задачи.
- Когда вы создаете проект, по умолчанию *MainTask* является непрерывной задачей. Вы можете оставить ее как есть, либо изменить ее свойства (имя, тип и т.д.).

См. *periodic task* (периодическая задача).

Control and Information Protocol (Протокол управления и информации)

Протокол передачи сообщений, используемый линией Logix5000 оборудования для систем управления от Allen-Bradley. Собственный протокол связи, используемый в сети ControlNet.

controller fault handler (обработчик ошибок контроллера)

Обработчик ошибок контроллера является дополнительной задачей, которая выполняется, когда:

- основная ошибка не является ошибкой выполнения инструкции
- процедура ошибки программы:
  - не смогла сбросить основную ошибку
  - дала ошибку
  - не существует

Вы можете создать только одну программу для обработчика ошибок контроллера. После того, как вы создадите эту программу, вам необходимо сконфигурировать одну процедуру в качестве главной процедуры.

- Программа ошибки контроллера *не* выполняет процедуру ошибки.
- Если вы зададите процедуру ошибки для программы ошибки контроллера, контроллер никогда не будет выполнять эту процедуру.
- Вы можете создать дополнительные процедуры и вызывать их из главной процедуры.

controller scope (область видимости контроллера)

Данные, доступные в любом месте контроллера. Контроллер содержит набор тегов, на которые могут ссылаться процедуры и теги-псевдонимы из любой программы, а также другие псевдонимы в области видимости контроллера.
См. *program scope* (область видимости программы).

Coordinated System Time (CST) (Согласованное системное время)

64-битовое значение, которое представляет собой количество микросекунд с момента, когда контроллер-задатчик CST начал отсчет.

- Значение CST сохраняется в виде массива DINT[2], где:
  - первый элемент сохраняет младшие 32 бита
  - второй элемент сохраняет старшие 32 бита
- Вы можете использовать временную метку CST для сравнения относительного времени разных выборок данных.

COUNTER
Структурный тип данных, содержащий информацию о статусе и управлении для инструкций счетчика.

D

data type (тип данных)
Определение размера и формата памяти, которые будут выделены, когда вы создадите тег этого типа данных.

decimal (десятичный)
Целочисленные значения, отображаемые и вводимые в десятичной системе счисления. Не имеют префикса. Не заполняют длину целочисленного значения. См. *binary* (двоичный), *hexadecimal* (шестнадцатеричный), *octal* (восьмеричный).

description (описание)
Дополнительный текст, который вы можете использовать для дальнейшего документирования вашего приложения.

- Вы можете использовать любые символы, которые можно вывести на печать, включая возврат каретки, табуляцию и пробел.
- Описания не загружаются в контроллер. Они остаются в оффлайновом файле проекта.
- Описания имеют следующие пределы по длине:
  - Для тегов вы можете использовать до 120 символов.
  - Для других объектов (задачи, программы, модули и т.д.) вы можете использовать до 128 символов.

dimension (измерение)
Задание размера массива. Массивы могут иметь максимум три измерения. См. *array* (массив).

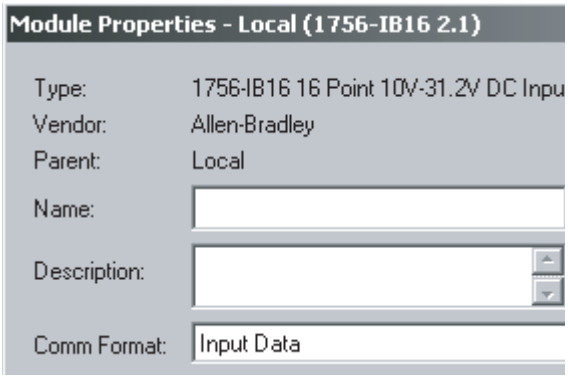
DINT
Тип данных, который сохраняет 32-битовое (4 байта) целочисленное значение со знаком (от -2,147,483,648 до +2,147,483,647). В контроллерах Logix5000 используйте значения DINT для целых чисел:

- Контроллеры Logix5000 выполняют свои функции более эффективно и используют меньше памяти, когда они работают с 32-битовыми целыми числами (DINT), а не с 16-битовыми целыми числами (INT) или 8-битовыми целыми числами (SINT).
- Обычно инструкции при выполнении преобразуют значения SINT или INT в **оптимальный тип данных** (обычно значение DINT или REAL). Минимизируйте использование типов данных SINT и INT, так как это требует дополнительных затрат времени и памяти.

direct connection (прямое соединение)

Прямое соединение – это соединение в реальном времени для передачи данных между контроллером и модулем ввода/вывода. Контроллер поддерживает и контролирует соединение с модулем ввода/вывода. Любое нарушение соединения, например, ошибка модуля или удаление модуля под напряжением, устанавливает биты ошибки в области данных, связанной с данным модулем.

Прямое соединение - это любое соединение, которое *не* использует Rack Optimization (Оптимизация по рэку) в качестве опции Comm Format.



См. *rack-optimized connection* (оптимизированное по рэку соединение).

disable keying (отмена ключа)

Режим защиты посредством электронного ключа, который не требует совпадения атрибутов физического модуля и модуля, сконфигурированного в программном обеспечении, в независимости от этого он устанавливает соединение с модулем. См. *compatible module* (совместимый модуль), *exact match* (точное соответствие).

download (загрузка)

Процесс передачи содержимого проекта на рабочей станции в контроллер. См. *upload* (выгрузка).

E

elapsed time (затраченное время)

Суммарное время, требуемое для выполнения всех операций, сконфигурированных внутри одной задачи.

- Если контроллер сконфигурирован для выполнения нескольких задач, затраченное время включает все время, используемое/совместно используемое другими задачами, выполняющими другие операции.
- В режиме онлайн вы можете использовать диалоговое окно *Task Properties* (Свойства задачи) для просмотра максимального времени сканирования и времени последнего сканирования в мс для текущей задачи. Эти значения представляют собой затраченное время, которое включает время, потраченное на ожидание выполнения задач с более высоким приоритетом.

См. *execution time* (время выполнения).

electronic keying (электронный ключ)
Функция модулей ввода/вывода линии 1756, с помощью которой можно запросить выполнение модулем электронной проверки, чтобы убедиться, что физический модуль соответствует модулю, который был сконфигурирован программным обеспечением. Позволяет пользователю с помощью программного обеспечения предотвращать нечаянное использование ненадлежащих модулей или ненадлежащих ревизий модулей. См. *compatible module* (совместимый модуль), *disable keying* (отмена ключа), *exact match* (точное совпадение).

element (элемент)

Адресуемая единица данных, являющаяся частью большей единицы данных. Одна единица массива.

- Вы задаете элемент в массиве с помощью нижних индексов:

Для такого массива:	Задайте:
одномерный	`array_name[subscript_0]`
двумерный	`array_name[subscript_0, subscript_1]`
трехмерный	`array_name[subscript_0, subscript_1, subscript_2]]`

См. *array* (массив).

exact match (точное совпадение)
Режим защиты посредством электронного ключа, который требует, чтобы все атрибуты (производитель, номер по каталогу, основная ревизия и неосновная ревизия) физического модуля и модуля, сконфигурированного в программном обеспечении, совпадали, чтобы установить соединение с модулем.

execution time (время выполнения)
Суммарное время, требуемое для выполнения одной программы.

- Время выполнения включает только время, которое используется одной программой, и не включает время, используемое/ совместно используемое программами других задач, выполняющими другие операции.
- В режиме онлайн вы можете использовать диалоговое окно *Program Properties* (Свойства программы) для просмотра максимального времени сканирования и времени последнего сканирования (в мс) для текущей задачи. Эти значения представляют собой времена выполнения данной программы, и не включают время, потраченное на ожидание выполнения задач с более высоким приоритетом.

См. *elapsed time* (затраченное время).

exponential (экспоненциальный)

Действительные значения, отображаемые и вводимые в научном или экспоненциальном формате. Число всегда отображается с одним знаком слева от десятичной точки, за которой следует десятичная часть и экспонента. См. *style* (стиль).

F

faulted mode (режим ошибки)

Контроллер сгенерировал основную ошибку, не смог сбросить ошибку и выключился.

См. *major fault* (основная ошибка).

float (плавающий)

Действительные значения, отображаемые и вводимые в формате с плавающей точкой. Количество знаков слева от десятичной точки варьируется в зависимости от величины числа. См. *style* (стиль).

H

hexadecimal (шестнадцатеричный)

Целочисленные значения, отображаемые и вводимые в шестнадцатеричной системе счисления (каждый разряд представляет собой 4 бита). Имеют префикс 16#. Заполняют длину булева или целочисленного значения (1, 8, 16 или 32 бита). При отображении каждая группа из четырех разрядов отделяется знаком подчеркивания для большей удобочитаемости. См. *binary* (двоичный), *decimal* (десятичный), *octal* (восьмеричный).

I

immediate value (непосредственное значение)

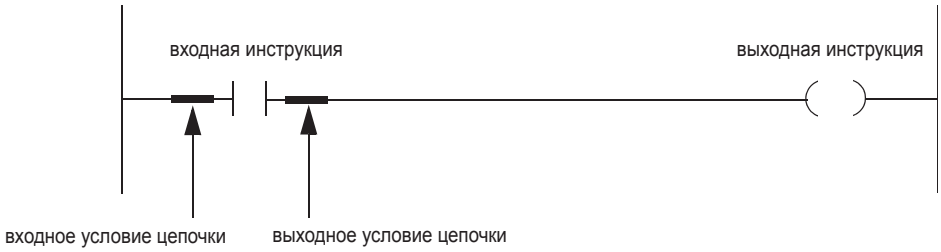
Фактическое 32-битовое действительное или целочисленное значение со знаком. Не является тегом, который сохраняет значение.

index (индекс)

Ссылка, используемая для указания на элемент внутри массива.

instruction (инструкция)

Контроллер оценивает инструкции релейной логики на основе условий цепочки, предшествующих инструкции (входное условие цепочки).



Только входные инструкции оказывают влияние на входное условие цепочки последующих инструкций в данной цепочке:

- Если входное условие цепочки для входной инструкции – «истина», то контроллер оценивает данную инструкцию и устанавливает выходное условие цепочки соответствующим результатам оценки.
  - Если результатом оценки инструкции является «истина», то выходное условие цепочки – «истина».
  - Если результатом оценки инструкции является «ложь», то выходное условие цепочки – «ложь».
- Выходная инструкция не изменяет выходного условия цепочки.
  - Если выходное условие цепочки для выходной инструкции – «истина», то выходное условие цепочки устанавливается на «истина».
  - Если выходное условие цепочки для выходной инструкции – «ложь», то выходное условие цепочки устанавливается на «ложь».

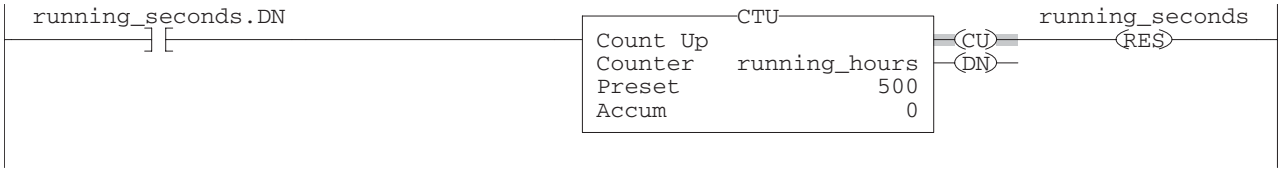
В контроллерах Logix5000 вы можете вводить несколько выходных инструкций на одну цепочку логики. Вы можете вводить выходные инструкции:

- одну за другой в цепочке (последовательно)
- между входными инструкциями, при условии, что последняя инструкция в цепочке является выходной

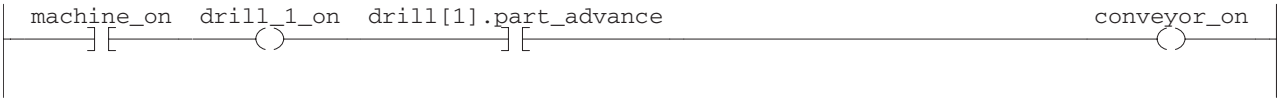
В следующем примере используется более одной выходной инструкции в цепочке.

ПРИМЕР Помещение нескольких выходных инструкций в цепочку.

Когда *running_secondsDN* включается, *running_hours* увеличивается на 1, а *running_seconds* сбрасывается.



Когда *machine_on* включается, то включается *drill_1_0*. Когда и *machine_on*, и *drill[1].part_advance* включены, включается *conveyor_on*.



INT

Тип данных, который сохраняет 16-битовое (2 байта) целочисленное значение (от -32,768 до +32,768). Минимизируйте использование этого типа данных:

- Как правило, инструкции преобразуют значения SINT или INT в **оптимальный тип данных** (обычно значение DINT или REAL) в процессе выполнения. Минимизируйте использование типов данных SINT и INT, так как это требует дополнительных затрат времени и памяти.

interface module (IFM) (интерфейсный модуль)

Предварительно подключенная ветвь полевой электрической цепи ввода/вывода.

L

listen-only connection (соединение только для прослушивания)

Соединение с модулем ввода/вывода, при котором другой контроллер владеет данными по конфигурации/предоставляет такие данные модулю ввода/вывода. Контроллер, использующий соединение только для прослушивания, не записывает данные по конфигурации и может поддерживать соединение с модулем ввода/вывода лишь в то время, когда контроллер-владелец активно управляет модулем ввода/вывода. См. *owner controller* (контроллер-владелец).

load (загрузить)

Скопировать проект из энергонезависимой памяти в пользовательскую память (RAM) контроллера. При этом будет перезаписан любой проект, находящийся в текущий момент в контроллере. См. *nonvolatile memory* (энергонезависимая память), *store* (сохранить).

M

main routine (главная процедура)

Первая процедура, которая должна выполняться при выполнении программы. Используйте главную процедуру для вызова (выполнения) других процедур (подпрограмм).

major fault (основная ошибка)

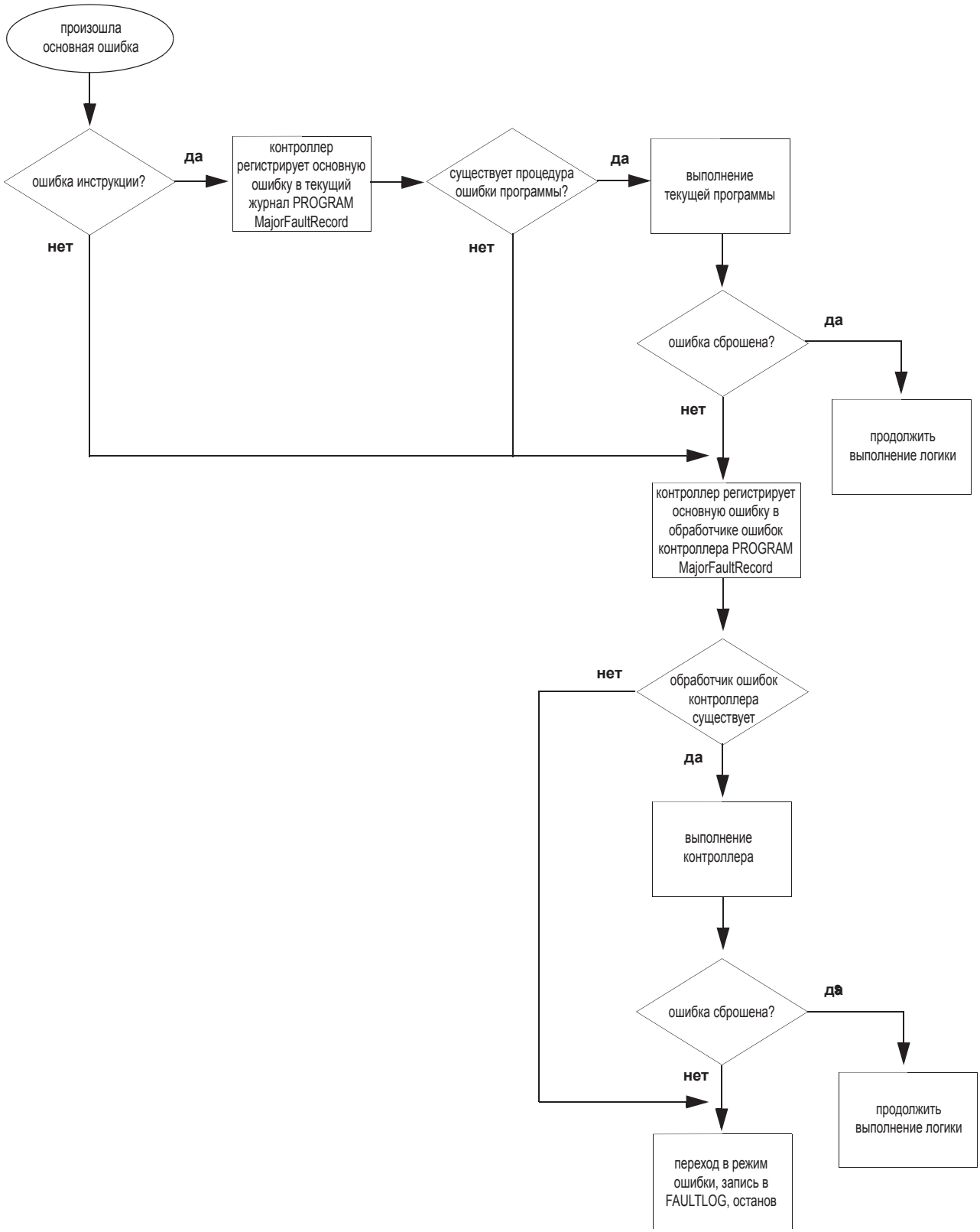
Состояние ошибки, являющееся достаточно серьезным для останова контроллера в том случае, если это состояние не будет сброшено. Когда происходит основная ошибка, контроллер:

1. Устанавливает бит основной ошибки.
2. Выполняет введенную пользователем логику ошибки, если она существует.
3. Если введенная пользователем логика ошибки не может сбросить ошибку, контроллер переходит в режим ошибки.
4. Устанавливает выходы в соответствии с их выходным состоянием в программном режиме.
5. Светодиодный индикатор подтверждения загорается красным.

Контроллер поддерживает два уровня обработки основных ошибок:

- процедура ошибки, содержащаяся в программе:
  - Каждая программа может иметь собственную процедуру ошибки.
  - Контроллер выполняет процедуру ошибки программы, когда происходит ошибка инструкции.
  - Если процедура ошибки программы не сбрасывает ошибку или если процедуры ошибки программы не существует, контроллер продолжает выполнение обработчика ошибок контроллера (если он задан).
- обработчик ошибок контроллера:
  - Если обработчика ошибок контроллера не существует или если он не может сбросить основную ошибку, контроллер входит в режим ошибки и выключается. При этом FAULLOG (журнал ошибок) обновляется. (См. следующую страницу).
  - Для всех ошибок, не связанных с инструкциями, т.е. ошибок ввода/вывода, сторожевого таймера задачи и т.д., сразу же выполняется обработчик ошибок контроллера. (Никакая процедура ошибки программы не вызывается).

Не сброшенная ошибка, а также не более двух дополнительных ошибок, которые не были сброшены ранее, регистрируются в журнале ошибок контроллера.


произошла основная ошибка
ошибка инструкции?
да
нет
контроллер регистрирует основную ошибку в текущий журнал PROGRAM MajorFaultRecord
существует процедура ошибки программы?
да
нет
выполнение текущей программы
ошибка сброшена?
да
нет
продолжить выполнение логики
контроллер регистрирует основную ошибку в обработчике ошибок контроллера PROGRAM MajorFaultRecord
обработчик ошибок контроллера существует
нет
да
выполнение контроллера
ошибка сброшена?
да
нет
продолжить выполнение логики
переход в режим ошибки, запись в FAULTLOG, останов

См. *faulted state* (состояние ошибки), *minor fault* (неосновная ошибка).

major revision (основная ревизия)

Линия модулей 1756 имеет указатели основной и неосновной ревизий. Основная ревизия обновляется каждый раз, когда в модуль вносится функциональное изменение. См. *electronic keying* (электронный ключ), *minor revision* (неосновная ревизия).

master (CST – согласованное системное время) (задатчик)

В пределах одного шасси, один и только один контроллер должен быть назначен задатчиком согласованного системного времени (CST). Все остальные модули в этом шасси синхронизируют свои значения CST со значением задатчика CST.

member (член)

Элемент структуры, имеющий собственные тип данных и имя.

- Члены могут также представлять собой структуры, создавая типы данных вложенной структуры.
- Члены внутри структуры могут иметь различные типы данных.
- Чтобы сделать ссылку на член в структуре, используйте этот формат:

  `tag_name.member_name` *(имя_тега.имя_члена)*

  Например:

Этот адрес:	Является ссылкой на:
timer_1.pre	Значение PRE (уставки) структуры *timer_1*.
input_load as data type (как тип данных) *load_info* *input_load.height*	Член *height* пользовательской структуры *input_load*.

- Если структура встроена в другую структуру, используйте имя тега структуры на самом высоком уровне, за которым следует имя тега подструктуры и имя члена:

  `tag_name.substructure_name.member_name`
  (имя_тега.имя_подструктуры.имя_члена)

  Например:

Этот адрес:	Является ссылкой на:
input_location as data type (как тип данных) *location* input_location.load_info.height	Член *height* структуры *load_info* в структуре *input_location*.

- Если структура определяет массив, используется тег массива, за которым следует позиция в массиве и имена подструктуры и члена.

  `array_tag[position].member` *(тег_массива[позиция].член)*

  или

  `array_tag[position].substructure_name.member_name`
  (тег_массива[позиция]имя_подструктурыимя_члена)

  Например:

Этот адрес:	Является ссылкой на:
conveyor[10].source	Член *source* 11-го элемента в массиве *conveyor* (индексация элементов массива начинается с нуля).
conveyor[10].info.height	Член *height* структуры *info* в 11-м элементе массива *conveyor* (индексация элементов массива начинается с нуля).

См. *structure* (структура).

memory (память)

Электронная запоминающая среда, встроенная в контроллер и используемая для хранения программ и данных.

minor fault (неосновная ошибка)

Состояние ошибки, которое не является достаточно серьезным для отключения контроллера:

Если происходит:	То контроллер:
проблема с инструкцией	1. 1. устанавливает S:MINOR 2. 2. регистрирует информацию об ошибке в атрибуте MinorFaultRecord объекта PROGRAM, 3. 3. устанавливает бит 4 атрибута MinorFaultRecord объекта FAULTLOG .
перекрытие периодической задачи	устанавливает бит 6 атрибута MinorFaultBits объекта FAULTLOG
проблема с последовательным портом	устанавливает бит 9 атрибута MinorFaultBits объекта FAULTLOG
разрядка батареи	устанавливает бит 10 атрибута MinorFaultBits объекта FAULTLOG

Чтобы сбросить неосновные ошибки:

1. В организаторе контроллера щелкните правой кнопкой мыши на папке *Controller* `name_of_controller` (имя контроллера) и выберите *Properties* (Свойства).
2. Нажмите на закладку *Minor Faults* (Неосновные ошибки).
3. Используйте информацию в списке *Recent Faults* (Последние ошибки) для исправления причины ошибки. Обратитесь к разделу «Коды неосновных ошибок» на стр. 16-4.

См. *major fault* (основная ошибка).

minor revision (неосновная ревизия)

Линия модулей 1756 имеет указатели основной и неосновной ревизий. Неосновная ревизия обновляется каждый раз, когда в модуль вносится изменение, не влияющее на его функции или интерфейс. См. *electronic keying* (электронный ключ), *major revision* (основная ревизия).

multicast (многоадресная передача)

Механизм, с помощью которого модуль может отсылать данные по сети так, что их получают одновременно несколько слушателей. Описывает свойство модуля ввода/вывода ControlLogix, позволяющее многочисленным контроллерам одновременно получать входные данные из одного модуля ввода/вывода.

multiple owners (многочисленные владельцы)

Настройка конфигурации, при которой несколько контроллеров имеют абсолютно одинаковую информацию о конфигурации, что позволяет им одновременно являться владельцами одного и того же модуля ввода.

N

name (имя)

Имена идентифицируют контроллеры, задачи, программы, теги, модули и т.д. Имена соответствуют правилам идентификатора IEC-1131-3 и:

- должны начинаться с буквы (A-Z или a-z) или знака подчеркивания (_)
- могут содержать только буквы, цифры и знаки подчеркивания
- могут включать не более 40 символов
- не должны включать несколько знаков подчеркивания (_) подряд или замыкающий знак подчеркивания (_)
- *не* чувствительны к регистру
- загружаются на контроллер

network update time (NUT) (время обновления сети)

Повторяющийся интервал времени, в течение которого данные могут быть отосланы по сети ControlNet. Время обновления сети имеет диапазон от 2 мс до 100 мс.

nonvolatile memory (энергонезависимая память)

Память контроллера, сохраняющая свое содержимое, когда контроллер отключен от сети или его батарея разряжена.

numeric expression (численное выражение)

В структурированном тексте это выражение, которое вычисляет целое значение или значение с плавающей точкой.

- Численное выражение использует арифметические операторы, арифметические функции и побитовые операторы. Например, tag1 + 5.
- Часто численное выражение вкладывается в булево выражение. Например, (tag1 + 5) > 65.

O

object (объект)

Структура данных, в которой хранится информация о состоянии. Когда вы вводите инструкцию GSV/SSV, вы задаете объект и его атрибут, к которому вы хотите обратиться. В некоторых случаях существует несколько экземпляров объекта одного и того же типа, в этой ситуации вам, возможно, придется также задать имя объекта. Например, в вашем приложении может быть несколько задач. Каждая задача имеет собственный объект TASK (задача), к которому вы обращаетесь по имени задачи.

octal (восьмеричный)

Целочисленные значения, отображаемые и вводимые в восьмеричной системе счисления (каждый разряд представляет собой 3 бита). Имеют префикс 8#. Заполняют длину булева или целочисленного значения (1, 8, 16 или 32 бита). При отображении каждая группа из трех разрядов отделяется знаком подчеркивания для большей удобочитаемости. См. *binary* (двоичный), *decimal* (десятичный), *hexadecimal* (шестнадцатеричный).

offline (оффлайн, автономный)

Просмотр и редактирование проекта, находящегося на жестком диске рабочей станции. См. online (онлайн).

online (онлайн)

Просмотр и редактирование проекта в контроллере. См. offline (оффлайн, автономный).

optimal data type (оптимальный тип данных)

Тип данных, который фактически использует инструкция Logix5000 (обычно это типы данных DINT и REAL).

- В справочных руководствах по набору инструкций оптимальный тип данных выделяется **жирным** шрифтом.
- Инструкция выполняется быстрее и требует меньше памяти, если все операнды инструкции используют:
  - один и тот же тип данных
  - оптимальный тип данных
- Если вы смешиваете типы данных и используете теги, имеющие не оптимальный тип данных, контроллер преобразует данные по следующим правилам
  - Являются ли *какие-либо* операнды значениями REAL?

Если:	То входные операнды (например, источник, тег в выражении, предел) преобразуются в:
Да	значения REAL
Нет	значения DINT

  - После выполнения инструкции результат (значение DINT или REAL) при необходимости преобразуется в тип данных приемника.

- Поскольку преобразование данных требует дополнительных затрат времени и памяти, вы можете увеличить эффективность ваших программ:
  - используя в инструкции один тип данных
  - минимально используя типы данных SINT или INT

Другими словами, используйте в ваших инструкциях, наряду с непосредственными значениями, все теги DINT или все теги REAL.

- В следующей таблице в сжатом виде описывается, как контроллер преобразует типы данных:

<table>
<tr><th>Преобразование:</th><th>Результат:</th></tr>
<tr><td>большее целое число в меньшее целое число</td><td>Контроллер отсекает верхнюю часть большего целого числа и генерирует переполнение.
Например:
<table>
<tr><th></th><th>Десятичное</th><th>Двоичное</th></tr>
<tr><td>DINT</td><td>65,665</td><td>0000_0000_0000_0001_0000_0000_1000_0001</td></tr>
<tr><td>INT</td><td>129</td><td>0000_0000_1000_0001</td></tr>
<tr><td>SINT</td><td>-127</td><td>1000_0001</td></tr>
</table></td></tr>
<tr><td>SINT или DINT в REAL</td><td>Точность данных не теряется.</td></tr>
<tr><td>DINT в REAL</td><td>Точность данных может быть потеряна. Оба типа данных сохраняют данные в 32-х битах, но тип REAL использует часть битов для хранения значения экспоненты. Если точность потеряна, контроллер восполняет ее за счет наименее значимой части DINT.</td></tr>
<tr><td>REAL в целое число</td><td>Контроллер округляет дробную часть и отсекает верхнюю часть недробной части. Если происходит потеря данных, контроллер устанавливает флаг состояния переполнения.
Числа округляются следующим образом:
• числа, отличные от x.5, округляются до ближайшего числа.
• X.5 округляются до ближайшего четного числа.
Например:
<table>
<tr><th>REAL (источник)</th><th>DINT (результат)</th></tr>
<tr><td>-2.5</td><td>-2</td></tr>
<tr><td>-1.6</td><td>-2</td></tr>
<tr><td>-1.5</td><td>-2</td></tr>
<tr><td>-1.4</td><td>-1</td></tr>
<tr><td>1.4</td><td>1</td></tr>
<tr><td>1.5</td><td>2</td></tr>
<tr><td>1.6</td><td>2</td></tr>
<tr><td>2.5</td><td>2</td></tr>
</table></td></tr>
</table>

overlap (перекрытие)

Состояние, при котором задача (периодическая или событийная) запускается в то время, когда еще выполняется задача, запущенная от предыдущего триггера.

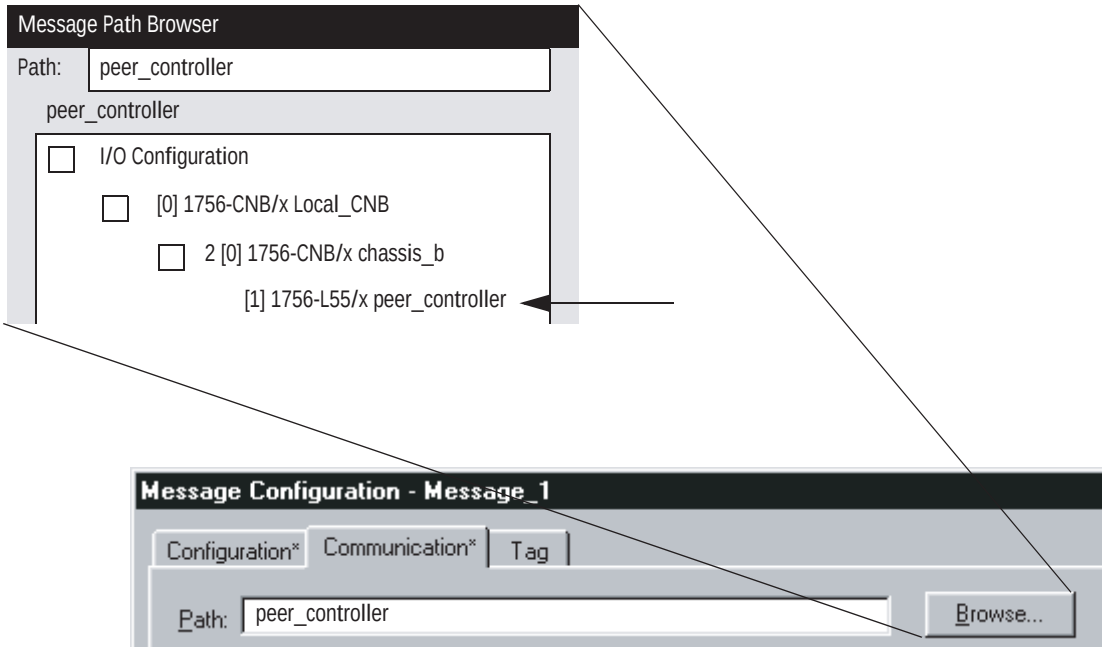
owner controller (контроллер-владелец)

Контроллер, который создает первичную конфигурацию и коммуникационное соединение с модулем. Контроллер-владелец записывает данные по конфигурации и может устанавливать соединение с модулем. См. *listen-only connection* (соединение только для прослушивания).

P

path (путь)

Путь описывает маршрут, по которому проходит сообщение, чтобы достичь получателя. Если конфигурация ввода/вывода контроллера содержит устройство-получателя, используйте кнопку *Browse* (Просмотр) для выбора этого устройства. Это автоматически определит путь.



Если конфигурация ввода/вывода *не* содержит устройства-получателя, то введите путь к получателю, используя следующий формат.

port, address, port, address

Где:	Для:	Является:
port	объединительной платы любого контроллера или модуля 1756	1
	порта DF1 контроллера Logix5000	2
	порта ControlNet модуля 1756-CNB	
	порта Ethernet модуля 1756-ENBx или –ENET	
	порта DH+ с каналом A модуля 1756-DHRIO	
	порта DH+ с каналом A модуля 1756-DHRIO	3
address	объединительной платы ControlLogix	номером слота
	сети DF1	адресом станции (0-254)
	сети ControlNet	номером узла (десятичное число 1-99)
	сети DH+	8#, за которым следует номер узла (восьмеричное число 1-77) Например, чтобы задать восьмеричный адрес узла 37, введите 8#37.
	сети EtherNet/IP	Вы можете задать модуль в сети EtherNet/IP, используя один из следующих форматов: IP адрес (например, 130.130.130.5) IP адрес:Порт (например, 130.130.130.5:24) имя DSN (например, tanks) имя DSN:Порт (например, tanks:24)

См. *connection* (соединение).

periodic task (периодическая задача)

Задача, которая запускается операционной системой с заданной периодичностью.

- Используйте периодические задачи для функций, требующих точного или детерминированного выполнения.
- Всякий раз по истечении заданного интервала времени задача запускается, и ее программы выполняются.
- Данные и выходные значения, полученные в результате выполнения программ задачи, остаются неизменными до следующего выполнения задачи, или же они управляются другой задачей.
- Вы можете конфигурировать период времени в пределах от 1мс до 2000 с. По умолчанию используется период времени 10 мс.

ВНИМАНИЕ

Убедитесь, что период времени длиннее, чем сумма времен выполнения всех программ, назначенных данной задаче. Если контроллер обнаружит возникновение триггера периодической задачи для задачи, которая уже выполняется, то произойдет неосновная ошибка.

- Периодические задачи всегда прерывают выполнение непрерывной задачи.
- В зависимости от уровня приоритета, периодическая задача может прерывать выполнение других периодических задач в контроллере.

См *continuous task* (непрерывная задача).

periodic task overlap (перекрытие периодической задачи)

Состояние, возникающее, когда при выполнении задачи такая же задача запускается снова. Время выполнения задачи больше периода, сконфигурированного для данной задачи. См. *periodic task* (периодическая задача).

predefined structure (предопределенная структура)

Структурный тип данных, который сохраняет информацию по определенной инструкции, например, структура TIMER относится к инструкциям таймера. Предопределенные структуры имеются всегда, независимо от конфигурации аппаратных средств системы. См. *product defined structure* (структура, определенная продуктом).

prescan (предварительное сканирование)

Предварительное сканирование – это промежуточное сканирование при переходе в режим Run (выполнения).

- Контроллер выполняет предварительное сканирование, когда вы переходите из режима Program (программы) в режим Run (выполнения).
- Предварительное сканирование проверяет все программы и инструкции и инициализирует данные на основе полученных результатов.
- Некоторые инструкции во время предварительного сканирования выполняются иначе, чем при обычном сканировании.

priority (приоритет)

Определяет, какая задача будет выполняться первой, если одновременно запускаются две задачи.

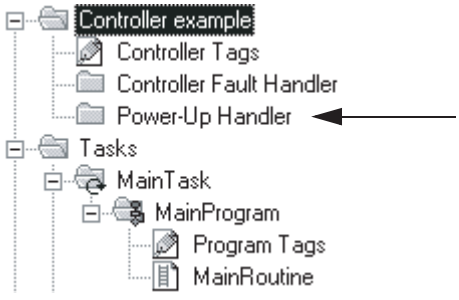
- Задача с более высоким приоритетом выполняется первой.
- Приоритеты находятся в диапазоне от1 до 15, где 1 – самый высокий приоритет.
- Задача с более высоким приоритетом будет прерывать все задачи, приоритет которых ниже.
- Если две задачи с одинаковым приоритетом запускаются одновременно, то контроллер переключается с одной задачи на другую каждую миллисекунду.

postscan (пост-сканирование)

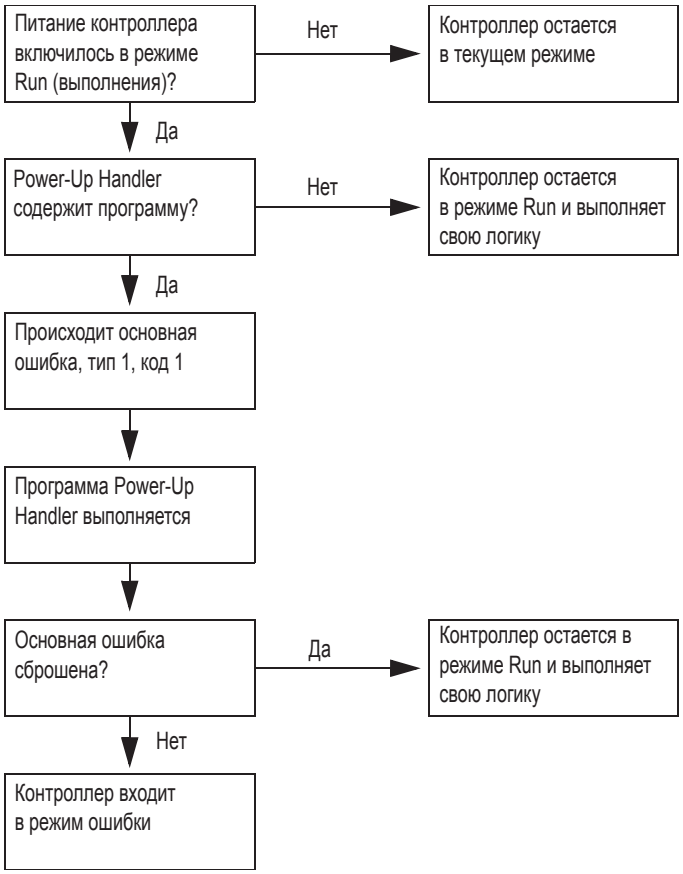
Функция контроллера, осуществляющая проверку логики программы перед запрещением программы с целью сброса инструкций и данных.

power-up handler (обработчик включения питания)

Дополнительная задача, которая выполняется, когда включается питание контроллера в режиме Run (выполнения). Чтобы использовать обработчик включения питания (Power-Up Handler), вы должны создать программу обработки включения питания и соответствующую главную процедуру.



Power-Up Handler (обработчик включения питания) выполняется следующим образом:



produced tag (производимый тег)

Тег, который контроллер делает доступным для использования другими контроллерами. Производимые теги всегда находятся в области видимости контроллера. См. *consumed tag* (потребляемый тег).

product defined structure (структура, определенная продуктом)

Структурный тип данных, который автоматически определяется программным обеспечением и контроллером. Конфигурируя модуль ввода/вывода, вы добавляете определенную продуктом структуру для этого модуля.

program (программа)
Набор взаимосвязанных процедур и тегов.

- Каждая программа содержит программные теги, главную выполняемую процедуру, другие процедуры, а также дополнительную процедуру обработки ошибок.
- Чтобы выполнить процедуры в программе, вы назначаете (планируете) программу для задачи:
  - Когда задача запускается, запланированные программы внутри задачи выполняются до полного завершения с первой до последней.
  - Когда задача выполняет программу, главная процедура программы выполняется первой.
  - Главная процедура может, в свою очередь, выполнять подпрограммы, используя инструкцию JSR.
- Папка *Unscheduled Programs* (Незапланированные программы) содержит программы, не назначенные задаче.
- Если логика в программе выдает основную ошибку, выполнение переходит к процедуре обработки ошибок, сконфигурированной для данной программы.
- Процедуры внутри программы могут обращаться к следующим тегам:
  - программные теги данной программы
  - теги контроллера
- Процедуры не могут обращаться к программным тегам других программ.

См. *routine* (процедура), *task* (задача).

program scope (область видимости программы)
Данные, доступные только внутри текущей программы. Каждая программа содержит набор тегов, на которые могут ссылаться только процедуры и теги-псевдонимы из этой программе. См. *controller scope* (область видимости контроллера).

project (проект)
Файл на вашей рабочей станции (или сервере), в котором хранятся логика, конфигурация, данные и документация для контроллера.

- Файл проекта имеет расширение .ACD.
- Когда вы создаете файл проекта, именем файла является имя контроллера.
- Имя контроллера не зависит от имени файла проекта. Если вы сохраните текущий файл проекта под другим именем, имя контроллера не изменится.
- Если имя контроллера отличается от имени файла проекта, в строке заголовка программного обеспечения RSLogix 5000 будут показаны оба имени.

См. *application* (приложение).

Q

qualifier (определитель)
В действиях последовательной функциональной схемы (ПФС) определитель задает момент начала и окончания действия.

См. *action* (действие), *sequential function chart* (последовательная функциональная схема), *step* (шаг).

R

rack-optimized connection (оптимизированное по рэку соединение)
Для цифровых модулей ввода/вывода вы можете выбрать оптимизированное по рэку соединение. Оптимизированное по рэку соединение консолидирует использование соединений между контроллером и всеми цифровыми модулями ввода/вывода в шасси (или на рейке стандарта DIN). Вместо отдельных прямых соединений для каждого модуля ввода/вывода получается одно соединение для всего шасси (или рейки DIN).

См. *direct connection* (прямое соединение).

rate (частота)
Для периодической задачи это частота, с которой контроллер выполняет задачу, от 1 мс до 2 000 000 мс (2000 секунд). По умолчанию используется 10 мс.

REAL
Тип данных, который сохраняет 32-битовые (4-байтные) значение с плавающей точкой стандарта IEEE в следующих диапазонах:

- от $-3.40282347E^{38}$ до $-1.17549435E^{-38}$ (отрицательные значения)
- 0
- от $1.17549435E^{-38}$ до $3.40282347E^{38}$ (положительные значения)

Тип данных REAL также сохраняет +-бесконечность, +-NAN и +-IND, но программное обеспечение отображает значения на экране по-разному в зависимости от формата дисплея.

Формат дисплея:	Эквивалент:	
Real (действительный)	+бесконечность	1.$
	-бесконечность	-1.$
	+NAN	1.#QNAN
	-NAN	-1.#QNAN
	-неопределенный	-1.#IND
Exponential (экспоненциальный)	+бесконечность	1.#INF000e+000
	-бесконечность	-1.#INF000e+000
	+NAN	1.#QNAN00e+000
	-NAN	-1.#QNAN00e+000
	-неопределенный	-1.#IND0000e+000

Программное обеспечение также сохраняет и отображает поднормальный диапазон стандарта IEEE:

- от -1.17549421E^{-38} до -1.40129846E^{-45} (отрицательные значения)
- от 1.40129846E^{-45} до 1.17549421E^{-38} (положительные значения)

removal and insertion under power (RIUP) (удаление и установка под напряжением)

Функция ControlLogix, позволяющая пользователю устанавливать и удалять модуль в то время, когда шасси находится под напряжением.

requested packet interval (RPI) (запрашиваемый межпакетный интервал)

При обмене данными по сети это максимальное время между каждой очередной выработкой входных данных.

- Как правило, этот интервал конфигурируется в микросекундах.
- Фактическая выработка данных ограничивается наибольшим кратным времени обновления сети, которое меньше RPI.
- Используйте значение, в два раза превышающее время обновления сети (network update power – NUT) ControlNet.

  Например, если NUT равен 5 мс, введите периодичность, равную 5, 10, 20, 40 мс и т.д.

См. *network update time (NUT)* (время обновления сети).

routine (процедура)

Набор логических инструкций на одном языке программирования, например, релейной логики.

- Процедуры предоставляют выполняемый код для проекта в контроллере (аналогично программному файлу для контроллера PLC или SLC).
- Каждая программа имеет главную процедуру:
  - Когда контроллер запускает связанную с ним задачу и выполняет соответствующую программу, главная процедура выполняется первой.
  - Чтобы вызвать другую процедуру внутри программы, введите инструкцию JSR в главную процедуру.
- Вы можете также задать дополнительную процедуру обработки ошибок в программе.
  - Если какая-либо из процедур в соответствующей программе выдает основную ошибку, контроллер выполняет процедуру обработки ошибки программы.

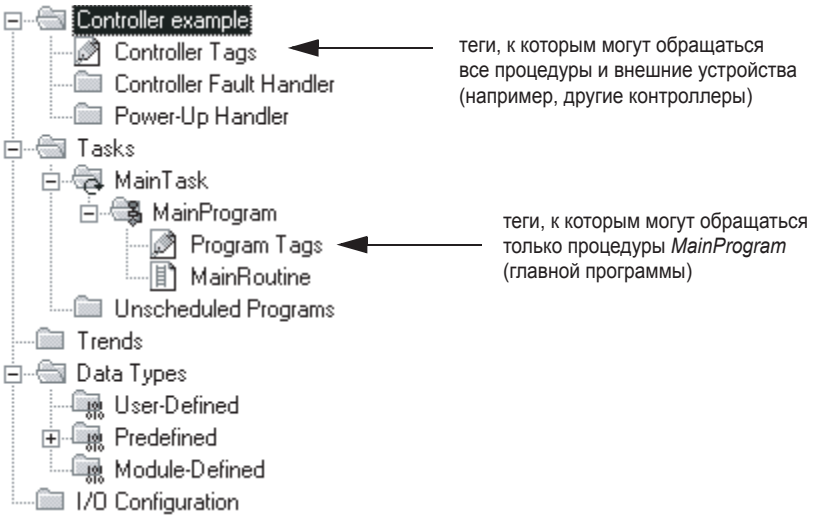
См. *program* (программа), *task* (задача).

scan time (время сканирования)

См. *elapsed time* (затраченное время), *execution time* (время выполнения).

scope (область видимости)

Определяет, откуда вы можете обращаться к определенному набору тегов. Когда вы создаете тег, вы назначаете его (задаете для него область видимости) тегом контроллера либо программным тегом для определенной программы, как показано ниже.



У вас может быть несколько тегов с одним именем:

- Все такие теги должны иметь разные области видимости. Например, один из этих тегов может быть тегом контроллера, а другие теги - программными тегами для разных программ. Либо все теги могут быть программными тегами разных программ.
- Внутри программы вы не можете ссылаться на тег контроллера, если тег с таким же именем существует в качестве программного тега для этой программы.

См. *controller scope* (область видимости контроллера), *program scope* (область видимости программы).

sequential function chart (последовательная функциональная схема)

Последовательная функциональная схема (ПФС) аналогична блок-схеме. В ней используются шаги и переходы для управления установкой или процессом.

См. *action* (действие), *step* (шаг), *transition* (переход),

SINT
Тип данных, который сохраняет 8-битовое (1-байтовое) целочисленное значение со знаком (от -128 до +127). Минимизируйте использование этого типа данных:

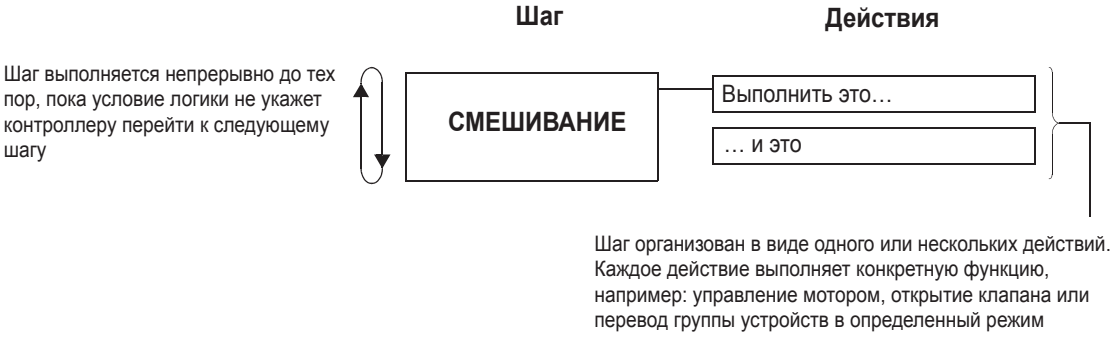
- Как правило, инструкции преобразуют значения SINT или INT в **оптимальный тип данных** (обычно это значение DINT или REAL) в процессе выполнения. Минимизируйте использование типов данных SINT и INT, так как это требует дополнительных затрат времени и памяти.

source key (ключ источника)
Механизм, ограничивающий число пользователей, которые могут просматривать процедуру.

- Вы можете назначить ключ источника одной или нескольким процедурам.
- Ключи источника подчиняются тем же правилам для имен, что и другие компоненты RSLogix 5000, а именно процедуры, теги и модули.
- Чтобы назначить ключ источника процедуре (защитить процедуру), используйте программное обеспечение RSLogix 5000. (Сначала вам нужно активизировать этот инструмент).
- Ключи источника сохраняются в файле ключей источника (sk.dat), отдельном от файлов проекта RSLogix 5000 (.acd).
- Чтобы просмотреть процедуру, защищенную ключом источника, вам необходимо иметь этот ключ.
- Без ключа источника вы не сможете открыть процедуру. Программное обеспечение RSLogix 5000 выведет на экран сообщение “Source Not Available” (Источник недоступен).
- Независимо от наличия или отсутствия ключа источника вы всегда можете загрузить проект и выполнить все процедуры.

См. *name* (имя).

step (шаг)
В последовательной функциональной схеме (ПФС) шаг представляет собой основную функцию процесса. Он содержит события, происходящие в определенное время, на определенной фазе или станции.



См. *action* (действие), *sequential function chart* (последовательная функциональная схема), *transition* (переход).

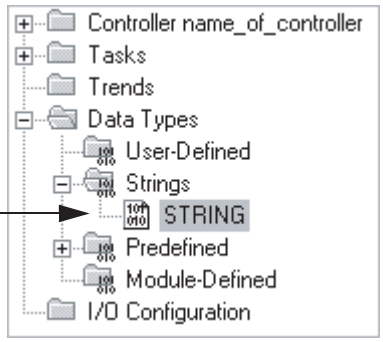
store (сохранить)

Скопировать проект в энергонезависимую память контроллера. Это перезапишет всякий проект, находящийся в текущий момент в энергонезависимой памяти.

См. *load* (загрузить), *nonvolatile memory* (энергонезависимая память).

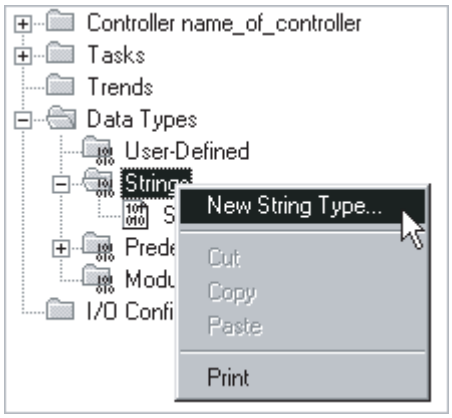
string (строка)

Группа типов данных, сохраняющих символы ASCII.



Вы можете использовать тип данных по умолчанию STRING. Он сохраняет до 82 символов

или



Вы можете создать новый строковый тип данных, который будет сохранять заданное вами количество символов

Каждый строковый тип данных содержит следующие члены:

Имя:	Тип данных:	Описание:	Примечания:
LEN	DINT	количество символов в строке	LEN автоматически обновляется до нового количества символов, всякий раз, когда вы: • используете диалоговое окно String Browser (Браузер строки) для ввода символов • используете инструкции, выполняющие считывание, преобразование строк или манипулирующие строками LEN показывает длину текущей строки. Член DATA может содержать дополнительные старые символы, не включенные в подсчет символов LEN.
DATA	массив SINT	символы ASCII строки	• Чтобы получить доступ к символам строки, обратитесь к имени тега. Например, чтобы получить доступ к символам тега *string_1*, введите *string_1*. • Каждый элемент массива DATA содержит один символ. • Вы можете создать новые строковые тип данных, которые будут сохранять большее или меньшее количество символов.

Новые строковые типы данных могут быть полезными в следующих ситуациях:

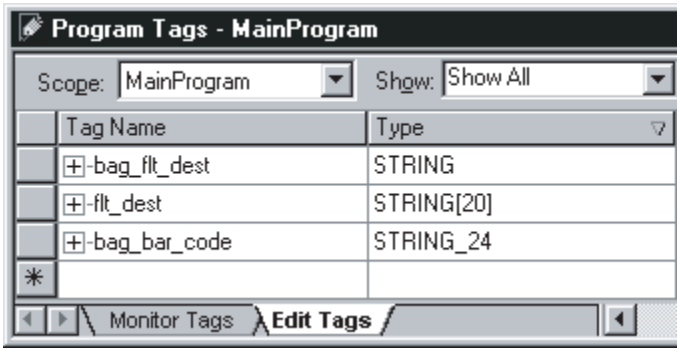
- Если вы имеете большое количество строк фиксированного размера, состоящих менее чем из 82 символов, вы можете сэкономить память, создав новый строковый тип данных.
- Если вам требуются строки, состоящие более чем из 82 символов, вы можете создать новый строковый тип данных, соответствующий необходимому количеству символов.

ВАЖНО

Будьте внимательны при создании нового строкового типа данных. Если вы впоследствии решите изменить размер строкового типа данных, вы можете потерять данные в тегах, которые в текущий момент используют этот тип данных.

Если вы:	То:
делаете строковый тип данных меньше	• • Данные усекаются. • • LEN остается неизменным.
делаете строковый тип данных больше	Данные и LEN сбрасываются на 0.

Следующий пример демонстрирует использование строкового типа данных STRING и нового строкового типа данных.



Этот тег использует тип данных STRING по умолчанию

Этот тег является массивом из 20 элементов, использующим тип данных STRING по умолчанию

Этот тег использует новый строковый тип данных.

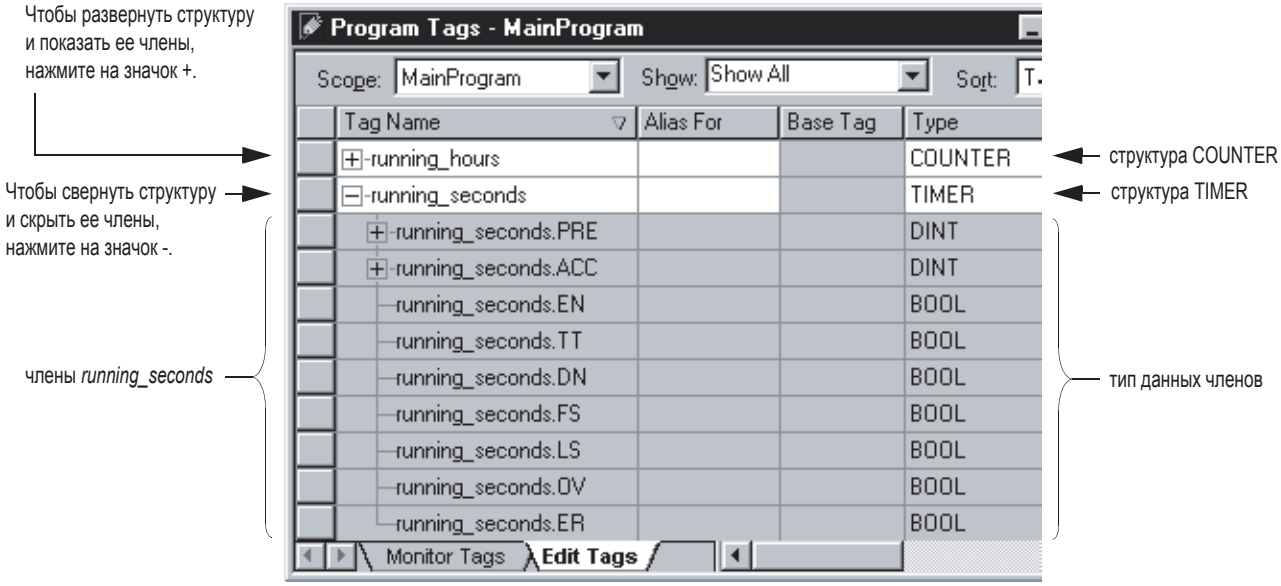
- Строковый тип данных, названный пользователем STRING_24.
- Новый строковый тип данных сохраняется.

structure (структура)

Некоторые типы данных представляют собой структуру.

- Структура сохраняет группу данных, при этом данные могут быть разных типов.
- Каждый отдельный тип данных внутри структуры называется **членом**.
- Как и теги, члены имеют имя и тип данных.
- Вы можете создавать собственные структуры, называемые **пользовательским типом данных**, различным образом комбинируя отдельные теги и большинство других структур.
- Чтобы скопировать данные в структуру, используйте инструкцию COP. Обратитесь к справочному руководству *Logix5000 Controllers General Instruction Set Reference Manual* (Справочное руководство по набору общих инструкций для контроллеров Logix5000), публикация 1756-RM003.

Типы данных COUNTER и TIMER являются примерами наиболее часто используемых структур.



См. *member* (член), *user-defined data type* (пользовательский тип данных).

style (стиль)

Формат, в котором отображаются числовые значения. См. *ASCII*, *binary* (двоичный), *decimal* (десятичный), *exponential* (экспоненциальный), *float* (плавающий), *hexadecimal* (шестнадцатеричный), *octal* (восьмеричный).

system overhead time slice (квант времени на служебные системные операции)

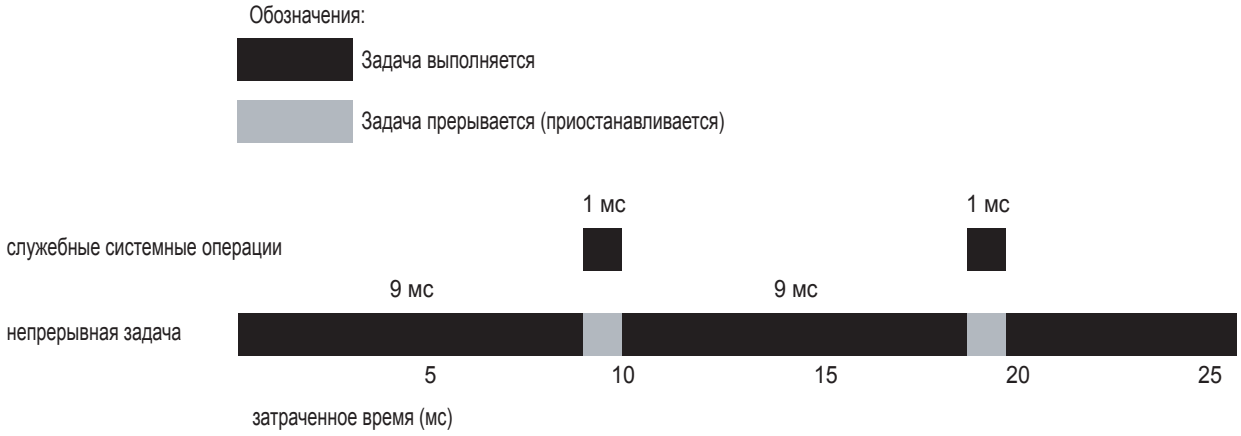
Задает процент времени контроллера (за исключением времени выполнения периодических задач), которое посвящено обмену данными и фоновым функциям (служебные системные операции):

- Контроллер выполняет служебные системные операции в течение максимум 1 мс за один раз.
- Если контроллер завершил выполнение служебных системных операций менее чем за 1 мс, он возобновляет выполнение непрерывной задачи.
- Обмен данными и фоновые функции включают следующее:
  - связь с программными устройствами и устройствами HMI (например, с программным обеспечением RSLogix 5000)
  - ответы на сообщения
  - отправку сообщений, включая пересылку блоков
  - восстановление и контроль соединений ввода/вывода (например, при удалении и установке модуля ввода/вывода при подключенном питании); сюда *не* входит обычный обмен ввода/вывода, осуществляемый при выполнении программы
  - связь с использованием моста между последовательным портом контроллера и другими устройствами ControlLogix через объединительную плату ControlLogix
- Если обмен данными осуществляется недостаточно быстро, увеличьте квант времени на служебные системные операции.

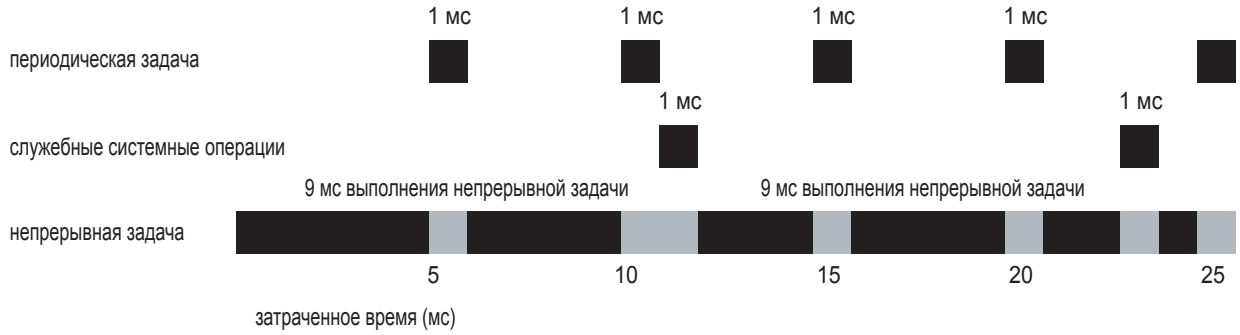
В следующей таблице показано отношение времени выполнения непрерывной задачи и служебных системных операций:

При таком кванте времени:	Непрерывная задача работает в течение:	А служебные системные операции выполняются в течение максимум:
10%	9 мс	1 мс
20%	4 мс	1 мс
33%	2 мс	1 мс
50%	1 мс	1 мс

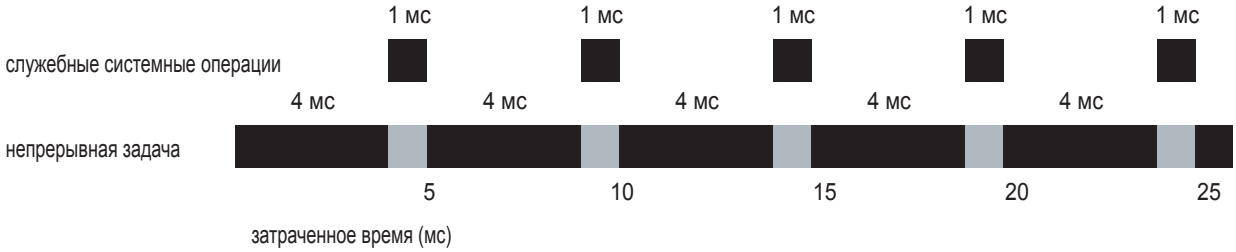
При используемом по умолчанию кванте времени 10% служебные системные операции прерывают выполнение непрерывной задачи каждые 9 мс (выполнения непрерывной задачи).



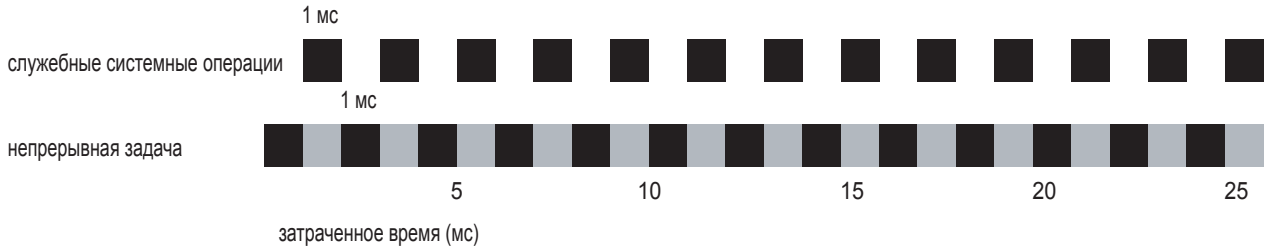
Прерывание выполнения периодической задачи увеличивает затраченное время (такт) между выполнениями служебных системных операций.



Если вы увеличите квант времени до 20%, служебные системные операции будут прерывать выполнение непрерывной задачи каждые 4 мс (выполнения непрерывной задачи).



Если вы увеличите квант времени до 50%, служебные системные операции будут прерывать выполнение непрерывной задачи каждую 1 мс (выполнения непрерывной задачи).



Если контроллер содержит только периодическую задачу (задачи), то значение кванта времени на служебные системные операции не будет оказывать никакого влияния. Служебные системные операции будут выполняться всякий раз, когда не будет работать периодическая задача.

Чтобы изменить квант времени на служебные системные операции:

1. Откройте проект RSLogix 5000.
2. В организаторе *контроллера щелкните правой кнопкой мыши на папке Controller* `name_of_controller` (имя контроллера) и выберите *Properties* (Свойства).
3. Нажмите на закладку *Advanced* (Расширенные).
4. В текстовом поле *System Overhead Time Slice* (Квант времени на служебные системные операции) введите или выберите процент времени на выполнение служебных системных операций (10-90%).
5. Нажмите *OK*.

T

tag (тег)

Именованная область памяти контроллера, где сохраняются данные.

- Теги являются основным механизмом распределения памяти, ссылки на данные из логики и осуществления контроля данных.
- Минимальный размер памяти, выделяемый для тега – четыре байта.
  - Когда вы создаете тег, сохраняющий типы данных BOOL, SINT или INT (которые имеют размер меньше 4 байтов), контроллер выделяет четыре байта, но данные заполняют только необходимую им часть этого пространства.
  - Пользовательские типы данных и массивы сохраняют данные в непрерывных блоках памяти и упаковывают типы данных меньшего размера в 32-битовые слова.

  Следующие примеры демонстрируют распределение памяти для различных тегов:

  - *start*, использующий тип данных BOOL:

Распределение памяти:	**Биты:**	
	31 ... **1**	**0**
распределение	не используются	start

  - *station_status*, использующий тип данных DINT:

Распределение памяти:	**Биты:**	
	31 ...	**0**
распределение	station_status	

  - *mixer*, использующий пользовательский тип данных:

Распределение памяти:	**Биты:**			
	31 ... **24**	**23** ... **16**	**15** ... **8**	**7** ... **0**
распределение 1	*mixer*.pressure			
распределение 2	*mixer*.temp			
распределение 3	*mixer*.agitate_time			
распределение 4	не используются	не используются	не используются	бит 0 *mixer.inlet* бит 1 *mixer.drain* бит 2 *mixer.agitate*

- *temp_buffer*, являющийся массивом из четырех значений INT (INT[4]):

Распределение памяти:	Биты:	
	31 **16**	**0**
распределение 1	*temp_buffer*[1]	*temp_buffer*[0]
распределение 2	*temp_buffer*[3]	*temp_buffer*[2]

См. *alias tag* (тег-псевдоним), *base tag* (базовый тег), *consumed tag* (потребляемый тег).

task (задача)
Механизм планирования выполнения программы.

- По умолчанию каждый новый файл проекта содержит предварительно сконфигурированную непрерывную задачу.
- При необходимости вы можете дополнительно сконфигурировать периодические задачи.
- Задача обеспечивает информацию о планировании и приоритетах для набора из одной или более программ, которые выполняются согласно определенным критериям.
- После того как задача запущена (активизирована), все программы, назначенные (запланированные) для этой задачи, выполняются в том порядке, в котором они отображаются в организаторе контроллера.
- Вы можете назначать программу только одной задаче за один раз.

См. *continuous task* (непрерывная задача), *periodic task* (периодическая задача).

timestamp (временная отметка)
Процесс ControlLogix, который записывает изменения во входных данных со ссылкой на относительное время, когда произошло соответствующее изменение.

transition (переход)
В последовательной функциональной схеме (ПФС) переход – это условие или условия «истина» или «ложь», которые определяют момент перехода к следующему шагу.

U

uncached connection (некэшированное соединение)
При использовании инструкции MSG некэшированное соединение указывает контроллеру закрыть соединение после завершения выполнения инструкции MSG. Сброс соединения оставляет его доступным для других нужд контроллера. См. *connection* (соединение), *cached connection* (кэшированное соединение).

unidirectional connection (однонаправленное соединение)

Соединение, в котором данные перемещаются только в одном направлении: от источника к получателю. См. *connection* (соединение), *bidirectional connection* (двунаправленное соединение).

unload (выгрузка)

Процесс перемещения содержимого контроллера в файл проекта на рабочей станции.

Если у вас нет файла проекта для контроллера, вы можете выгрузить данные из контроллера и создать файл проекта. Однако не все, что сохраняется в файле проекта, вы можете получить из контроллера. Если вы выгружаете данные из контроллера, новый файл проекта не будет содержать:

- комментарии цепочек
- описания для тегов, задач, программ, процедур, модулей и пользовательских структур
- цепочки псевдонимов (псевдонимы, указывающие на другие псевдонимы)

  Цепочки псевдонимов невозможно полностью воссоздать из контроллера. Если для элемента данных существует несколько возможных имен, микропрограммное обеспечение и программное обеспечение выберут наиболее подходящий псевдоним, который может не отражать то, как этот псевдоним был задан в оригинальном проекте.

См. *download* (загрузка).

user-defined data type (пользовательский тип данных)

Вы также можете создавать собственные **структуры**, называемые «пользовательский тип данных» (также часто называемые «пользовательские структуры»). Пользовательский тип данных группирует различные типы данных в один именованный объект.

- Внутри пользовательского типа данных вы должны определить **члены**.
- Как и теги, члены имеют имя и тип данных.
- Вы можете включать массивы и структуры.
- После создания пользовательского типа данных вы можете создать один или несколько тегов с использованием этого типа данных.
- Минимизируйте использование следующих типов данных, поскольку они обычно увеличивают объем потребной памяти и время выполнения логики:
  - **INT**
  - **SINT**

Например, некоторые системные значения используют тип данных SINT или INT. Если вы создаете пользовательский тип данных для сохранения этих значений, используйте соответствующий тип данных SINT или INT.

- Если вы включаете члены, представляющие устройства ввода/вывода, вам необходимо использовать релейную логику для копирования данных из членов структуры в соответствующие теги ввода/вывода и обратно. Обратитесь к разделу «Буферизация ввода/вывода» на стр. 2-8.
- Когда вы используете типы данных BOOL, SINT и INT, располагайте члены, использующие один и тот же тип данных, последовательно:

более эффективно	**менее эффективно**
BOOL	BOOL
BOOL	DINT
BOOL	BOOL
DINT	DINT
DINT	BOOL

- Вы можете использовать одномерные массивы.
- Вы можете создавать, редактировать и удалять пользовательские типы данных, только при программировании в режиме оффлайн.
- Если вы модифицируете пользовательский тип данных и измените его размер, существующие значения тегов, использующих этот тип данных, установятся на ноль (0).
- Чтобы скопировать данные в структуру, используйте инструкцию COP. Обратитесь к справочному руководству *Logix5000 Controllers General Instruction Set Reference Manual* (Справочное руководство по набору общих инструкций для контроллеров Logix5000), публикация 1756-RM003.

См. *structure* (структура).

W

watchdog (сторожевой таймер)

Определяет, через какое время работы задачи будет инициирована основная ошибка.

- В каждой задаче имеется сторожевой таймер, контролирующий выполнение задачи.
- Время сторожевого таймера может варьироваться от 1 мс до 2000000 мс (2000 секунд). Значение по умолчанию – 500 мс.
- Сторожевой таймер начинает отсчет времени при запуске задачи и останавливается, когда будут выполнены все программы в данной задаче.
- Если время выполнения задачи превышает время сторожевого таймера (контрольное время), возникает основная ошибка. (Время включает прерывания другими задачами).
- Ошибка по превышению контрольного времени (основная ошибка) также возникает при повторном запуске задачи в процессе ее выполнения (перекрытие задач). Это может произойти в том случае, если задача прерывается более высокоприоритетной, что задерживает выполнение задачи с более низким приоритетом.
- Вы можете сбросить ошибку превышения контрольного времени при помощи обработчика ошибок контроллера. Если при сканировании той же самой логики вновь возникнет такая же ошибка превышения контрольного времени, контроллер перейдет в режим ошибки независимо от того, сбросил ли обработчик ошибок контроллера эту ошибку превышения контрольного времени.

ВНИМАНИЕ

При достижении сторожевым таймером конфигурируемой уставки времени возникает основная ошибка. В зависимости от обработчика ошибок контроллера может произойти останов контроллера.

Чтобы изменить контрольное время для задачи:

1. Откройте проект RSLogix 5000.
2. В организаторе контроллера щелкните правой кнопкой мыши по соответствующей задаче и выберите *Properties* (Свойства).
3. Нажмите на закладку *Configuration* (Конфигурация).
4. В текстовом окне *Watchdog* (Сторожевой таймер) введите контрольное время.

Нажмите *OK*.

Числовой

A

B

C

D

E

F

G

I

N

O

P

T

U

V

W

How Are We Doing?

Your comments on our technical publications will help us serve you better in the future. Thank you for taking the time to provide us feedback.

You can complete this form and mail it back to us, visit us online at www.ab.com/manuals, or email us at RADocumentComments@ra.rockwell.com

vr

Pub. Title/Type Logix5000™ Controllers Common Procedures

Cat. No. 1756 ControlLogix®, 1769 CompactLogix™, 1789 SoftLogix™, 1794 FlexLogix™, PowerFlex 700S with DriveLogix

Pub. No. 1756-PM001G-EN-P Pub. Date March 2004 Part No. 957867-41

Please complete the sections below. Where applicable, rank the feature (1=needs improvement, 2=satisfactory, and 3=outstanding).

Overall Usefulness	1	2	3	How can we make this publication more useful for you?
Completeness (all necessary information is provided)	1	2	3	Can we add more information to help you? procedure/step illustration feature example guideline other explanation definition
Technical Accuracy (all provided information is correct)	1	2	3	Can we be more accurate? text illustration
Clarity (all provided information is easy to understand)	1	2	3	How can we make things clearer?
Other Comments				You can add additional comments on the back of this form.

Your Name _____ Location/Phone _____

Your Title/Function _____

Would you like us to contact you regarding your comments?

___No, there is no need to contact me

___Yes, please call me

___Yes, please email me at _____

___Yes, please contact me via _____

Return this form to: Allen-Bradley Marketing Communications, 1 Allen-Bradley Dr., Mayfield Hts., OH 44124-9705

Phone: 440-646-3176 Fax: 440-646-3525 Email: RADocumentComments@ra.rockwell.com

PLEASE FASTEN HERE (DO NOT STAPLE)

Other Comments

PLEASE FOLD HERE

PLEASE REMOVE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 18235 CLEVELAND OH

POSTAGE WILL BE PAID BY THE ADDRESSEE

Allen-Bradley
RELIANCE ELECTRIC
DODGE
ROCKWELL SOFTWARE
Rockwell Automation

1 ALLEN-BRADLEY DR
MAYFIELD HEIGHTS OH 44124-9705

Коды символов ASCII

Символ	Десят.	Шестн.	Символ	Десят.	Шестн.	Символ	Десят.	Шестн.	Символ	Десят.	Шестн.
[ctrl-@] NUL	0	$00	SPACE	32	$20	@	64	$40	‘	96	$60
[ctrl-A] SOH	1	$01	!	33	$21	A	65	$41	a	97	$61
[ctrl-B] STX	2	$02	“	34	$22	B	66	$42	b	98	$62
[ctrl-C] ETX	3	$03	#	35	$23	C	67	$43	c	99	$63
[ctrl-D] EOT	4	$04	$	36	$24	D	68	$44	d	100	$64
[ctrl-E] ENQ	5	$05	%	37	$25	E	69	$45	e	101	$65
[ctrl-F] ACK	6	$06	&	38	$26	F	70	$46	f	102	$66
[ctrl-G] BEL	7	$07	‘	39	$27	G	71	$47	g	103	$67
[ctrl-H] BS	8	$08	(	40	$28	H	72	$48	h	104	$68
[ctrl-I] HT	9	$09	)	41	$29	I	73	$49	i	105	$69
[ctrl-J] LF	10	$I ($0A)	*	42	$2A	J	74	$4A	j	106	$6A
[ctrl-K] VT	11	$0B	+	43	$2B	K	75	$4B	k	107	$6B
[ctrl-L] FF	12	$0C	,	44	$2C	L	76	$4C	l	108	$6C
[ctrl-M] CR	13	$r ($0D)	-	45	$2D	M	77	$4D	m	109	$6D
[ctrl-N] SO	14	$0E	.	46	$2E	N	78	$4E	n	110	$6E
[ctrl-O] SI	15	$0F	/	47	$2F	O	79	$4F	o	111	$6F
[ctrl-P] DLE	16	$10	0	48	$30	P	80	$50	p	112	$70
[ctrl-Q] DC1	17	$11	1	49	$31	Q	81	$51	q	113	$71
[ctrl-R] DC2	18	$12	2	50	$32	R	82	$52	r	114	$72
[ctrl-S] DC3	19	$13	3	51	$33	S	83	$53	s	115	$73
[ctrl-T] DC4	20	$14	4	52	$34	T	84	$54	t	116	$74
[ctrl-U] NAK	21	$15	5	53	$35	U	85	$55	u	117	$75
[ctrl-V] SYN	22	$16	6	54	$36	V	86	$56	v	118	$76
[ctrl-W] ETB	23	$17	7	55	$37	W	87	$57	w	119	$77
[ctrl-X] CAN	24	$18	8	56	$38	X	88	$58	x	120	$78
[ctrl-Y] EM	25	$19	9	57	$39	Y	89	$59	y	121	$79
[ctrl-Z] SUB	26	$1A	:	58	$3A	Z	90	$5A	z	122	$7A
ctrl-[ESC	27	$1B	;	59	$3B	[	91	$5B	{	123	$7B
[ctrl-\] FS	28	$1C	<	60	$3C	\	92	$5C	|	124	$7C
ctrl-] GS	29	$1D	=	61	$3D	]	93	$5D	}	125	$7D
[ctrl-^] RS	30	$1E	>	62	$3E	^	94	$5E	~	126	$7E
[ctrl-_] US	31	$1F	?	63	$3F	_	95	$5F	DEL	127	$7F

Поддержка Rockwell Automation

Rockwell Automation предоставляет техническую информацию в Интернете, чтобы помочь вам в использовании наших продуктов. По адресу http://support.rockwellautomation.com вы найдете технические руководства, базу FAQ, технические заметки и заметки по использованию, образец программного кода и ссылки на пакеты обновления программного обеспчечения, а также средство MySupport, которое вы можете настроить под себя для наилучшего использования этих инструментов.

Дополнительно к технической поддержке по телефону при установке, конфигурировании и поиске неисправностей, мы предлагаем программы TechConnect Support. За более подробной информацией вы можете обратиться к местным дистрибьюторам, представителям Rockwell Automation или на сайт http://support.rockwellautomation.com.

Помощь при установке

Если у вас есть проблемы с аппаратным модулем в первые 24 часа после установки, пожалуйста, обратитесь к данному руководству. Вы также можете обратиться в службу поддержки пользователей за первой помощью по наладке и запуску вашего модуля по телефону:

США	1.440.646.3223 Понедельник – пятница, 8:00 – 17:00 по восточному времени
Для других стран	Пожалуйста, обращайтесь к местным представителям Rockwell Automation за любой технической поддержкой.

Возврат новых продуктов в связи с неудовлетворительной работой

Компания Rockwell проверяет все свои продукты на предмет полной работоспособности после отгрузки с завода-изготовителя. Однако, если ваш продукт не работает в полном объеме и должен быть возвращен:

США	Обратитесь к вашему дистрибьютору. Для организации возврата вы должны сообщить дистрибьютору номер, присвоенный службой поддержки пользователей (Customer Support) (его можно получить по указанному выше номеру телефона).
Для других стран	Обратитесь к местному представителю Rockwell Automation для организации процедуры возврата.

www.rockwellautomation.com

Corporate Headquarters
Rockwell Automation, 777 East Wisconsin Avenue, Suite 1400, Milwaukee, WI, 53202-5302 USA, Tel: (1) 414.212.5200, Fax: (1) 414.212.5201

Headquarters for Allen-Bradley Products, Rockwell Software Products and Global Manufacturing Solutions
Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe: Rockwell Automation SA/NV, Vorstlaan/Boulevard du Souverain 36-BP 3A/B, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Headquarters for Dodge and Reliance Electric Products
Americas: Rockwell Automation, 6040 Ponders Court, Greenville, SC 29615-4617 USA, Tel: (1) 864.297.4800, Fax: (1) 864.281.2433
Europe: Rockwell Automation, Brühlstraße 22, D-74834 Elztal-Dallau, Germany, Tel: (49) 6261 9410, Fax: (49) 6261 17741
Asia Pacific: Rockwell Automation, 55 Newton Road, #11-01/02 Revenue House, Singapore 307987, Tel: (65) 351 6723, Fax: (65) 355 1733

Publication 1756-PM001G-EN-P - March 2004 PN 957867-41
Supersedes Publication 1756-PM001F-EN-P - June 2003

Связывайтесь с нами теперь по адресу www.rockwellautomation.com

Всякий раз, когда вы нуждаетесь в нас, Rockwell Automation осуществляет комплексное использование ведущих марок в промышленной автоматике, включающих элементы управления Allen-Bradley, продукты подачи питания Reliance Electric, механические компоненты подачи питания Dodge и Rockwell Software. Уникальный гибкий подход Rockwell Automation в помощи клиентам достигнуть конкурентного преимущества поддерживается тысячами авторизованными партнерами, дистрибьюторами и системными интеграторами по всему миру.

Представительство Rockwell Automation в Москве: 113054, Москва, Большой Строченовский пер., 22/25, Офис 402
Телефон: (095)956-0464, (095)956-0465; Факс: (095)956-0469; E-mail: software@rockwell.ru, info@rockwell.ru

Americas Headquarters, 1201 South Second Street, Milwaukee, WI 53204, USA, Tel: (1) 414 382-2000, Fax: (1) 414 382-4444
European Headquarters SA/NV, avenue Herrmann Debroux, 46, 1160 Brussels, Belgium, Tel: (32) 2 663 06 00, Fax: (32) 2 663 06 40
Asia Pacific Headquarters, 27/F Citicorp Centre, 18 Whitfield Road Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 1756-PM001G-EN-P - March 2004