Allen-Bradley

Языки программирования последовательной функциональной схемы (ПФС) и структурированного текста (СТ)

Выдержка из публикации 1756-PM001 *"Общая методика для программируемых контроллеров Logix5000"*

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.

Назначтение данного руководства

Данный документ представляет собой выдержку из публикации 1756-PM001 *"Общая методика для программируемых контроллеров Logix5000"*. Он содержит общие для всех контроллеров Logix5000 пошаговые методики выполнения следующих задач:

- Разработка, программирование и форсировка последовательной функциональной схемы
- Программирование процедур с использованием языка программирования структурированного текста

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

- Контроллеры CompactLogix™
- Контроллеры ControlLoxic®
- Контроллеры FlexLogix™
- PowerFlex® 700S с контроллерами DriveLogix
- Контроллеры SoftLogix5800™

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.

Разработка последовательной функциональной схемы

Глава 5

Когда использовать эту процедуру 5-1
Как использовать эту процедуру 5-1
Что такое последовательная функциональная схема? 5-2
Как разработать ПФС: Обзор 5-4
Определение задач 5-5
Выбор способа выполнения ПФС 5-6
Определение шагов процесса 5-6
- Руководящие указания 5-7
- Структура SFC_STEP 5-8

Организация шагов 5-12
- Обзор 5-12
- Последовательность 5-14
- Ветвь выбора 5-15
- Одновременная ветвь 5-16
- Связь, ведущая к предыдущему шагу 5-17

Добавление действий для каждого шага 5-18
- Как вы хотите использовать действие? 5-18
- Использование небулева действия 5-18
- Использование булева действия 5-20
- Структура SFC_ACTION 5-20

Описание каждого действия в псевдокоде 5-21
Выбор определителя для действия 5-23
Задание условий перехода 5-24
- Тег перехода 5-26
- Как вы хотите запрограммировать переход? 5-26
- Использование выражения BOOL 5-26
- Вызов подпрограммы 5-27

Переход по истечении заданного времени 5-28
Выключение устройства в конце шага 5-32
- Выбор опции последнего сканирования 5-32
- Использование опции Don't Scan (Не сканировать) 5-34
- Использование опции Programmatic Reset (Программный сброс) 5-35
- Использование опции Automatic Reset (Автоматический сброс) 5-38

Поддержание чего-либо во включенном состоянии от шага к шагу 5-40
- Как вы хотите управлять устройством? 5-40
- Использование одновременной ветви 5-41
- Сохранение и сброс действия 5-42
- Использование одного укрупненного шага 5-44

Окончание ПФС 5-45
- Что вы хотите делать в конце ПФС? 5-45
- Использование стопового элемента 5-45
- Перезапуск (сброс) ПФС 5-46
- Структура SFC_STOP 5-47

Вложение ПФС 5-49
- Передача параметров 5-50

Конфигурирование момента возврата к OC/JSR 5-50
Приостановка или сброс ПФС 5-51
Схемы выполнения 5-51

Программирование последовательной функциональной схемы

Глава 6

Когда использовать данную процедуру ... 6-1
Перед началом использования данной процедуры ... 6-1
Как использовать данную процедуру ... 6-2
Добавление элемента ПФС (Add an SFC Element) ... 6-3
Добавить и вручную подсоединить элементы ... 6-3
Добавить и автоматически подсоединить элементы ... 6-4
Добавить элементы методом буксировки ... 6-4
Создание совместной ветви ... 6-5
Начало совместной ветви ... 6-5
Окончание совместной ветви ... 6-5
Создание ветви выбора ... 6-6
Начало ветви выбора ... 6-6
Окончание ветви выбора ... 6-7
Настройки приоритетов ветви выбора ... 6-8
Возвращение на предыдущий шаг ... 6-9
Подключение связи к шагу ... 6-9
Скрытая связь ... 6-10
Вывод на экран скрытой связи ... 6-10
Изменение имени шага ... 6-11
Конфигурирование шага ... 6-11
Присвоение шагу заданного времени ... 6-11
Конфигурирование сигналов тревоги для шага ... 6-12
Использование выражения для расчета времени ... 6-12
Переименование перехода ... 6-14
Программирование перехода ... 6-14
Ввод булева выражения ... 6-14
Вызов подпрограммы ... 6-15
Добавление операции ... 6-16
Переименование операции ... 6-16
Конфигурирование операции ... 6-17
Изменение управляющего параметра операции ... 6-17
Расчет заданного времени при выполнении ... 6-18
Обозначение операции как булевой операции ... 6-19
Программирование операции ... 6-19
Ввод структурированного текста ... 6-19
Вызов подпрограммы ... 6-21
Присваивание порядка выполнения операции ... 6-22
Документирование ПФС ... 6-23
Добавить комментарии на языке структурированного текста ... 6-23
Добавить описания тега ... 6-24
Добавить текстовое окно ... 6-25
Показать или спрятать текстовое окно или описание тегов ... 6-26
Показать или спрятать текстовые окна или описания ... 6-26
Спрятать описание отдельного тега ... 6-27
Конфигурирование выполнения ПФС ... 6-28
Проверка процедуры ... 6-29

Глава 7

Программирование на языке структурированного текста

Когда пользоваться данной главой 7-1
Синтаксис языка структурированного текста 7-1
Присваивание 7-2
Задание присваивания без сохранения 7-3
Присваивание символов ASCII строке 7-4
Выражения 7-4
Использование арифметических операторов и функций 7-6
Использование операторов отношения 7-7
Как производятся операции со строками 7-8
Использование логических операторов 7-9
Использование поразрядных операторов 7-10
Определение порядка выполнения 7-10
Инструкции 7-11
Конструкции 7-12
IF...THEN 7-13
CASE...OF 7-16
FOR...DO 7-19
WHILE...DO 7-22
REPEAT...UNTIL 7-25
Комментарии 7-28

Глава 14

Форсировка элементов релейной логики

Когда использовать данную процедуру 14-1
Как использовать данную процедуру 14-1
Меры предосторожности 14-2
Разрешение форсировок 14-2
Запрещение или удаление форсировки 14-3
Проверка состояния форсировок 14-4
Панель инструментов Online 14-4
Светодиод FORCE 14-5
Инструкция GSV 14-5
Что форсировать 14-6
Когда использовать форсировку ввода\вывода 14-6
Форсировка входного значения 14-7
Форсировка выходного значения 14-7
Добавление форсировки ввода\вывода 14-8
Когда использовать проход 14-9
Проход через переход или форсировку пути 14-9
Когда использовать форсировку ПФС 14-9
Форсировка перехода 14-9
Форсировка параллельного пути 14-11
Добавить форсировку ПФС 14-12
Удаление или запрещение форсировок 14-13
Удаление отдельной форсировки 14-13
Запрещение всех форсировок ввода/вывода 14-14
Удаление всех форсировок ввода/вывода 14-14
Запрещение всех форсировок ПФС 14-14
Удаление всех форсировок ПФС 14-14

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

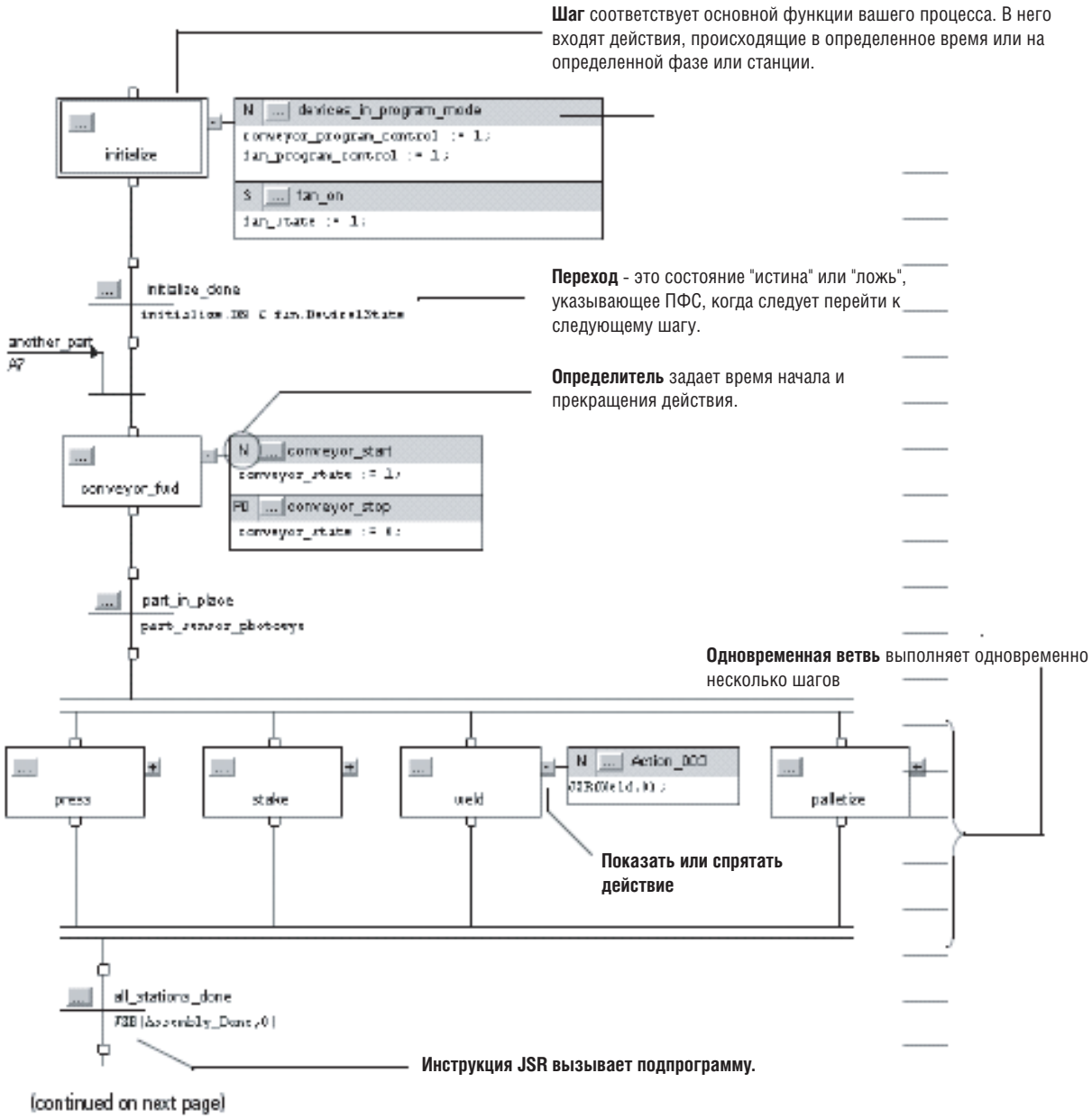
Как правило, разработка ПФС представляет собой итерационный процесс. По желанию вы можете использовать программное обеспечение RSLogix5000 для проектирования и доработки ПФС. Конкретные процедуры ввода ПФС приводятся в разделе «Программирование последовательной функциональной схемы» на стр. 6-1.

Что такое последовательная функциональная схема?

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

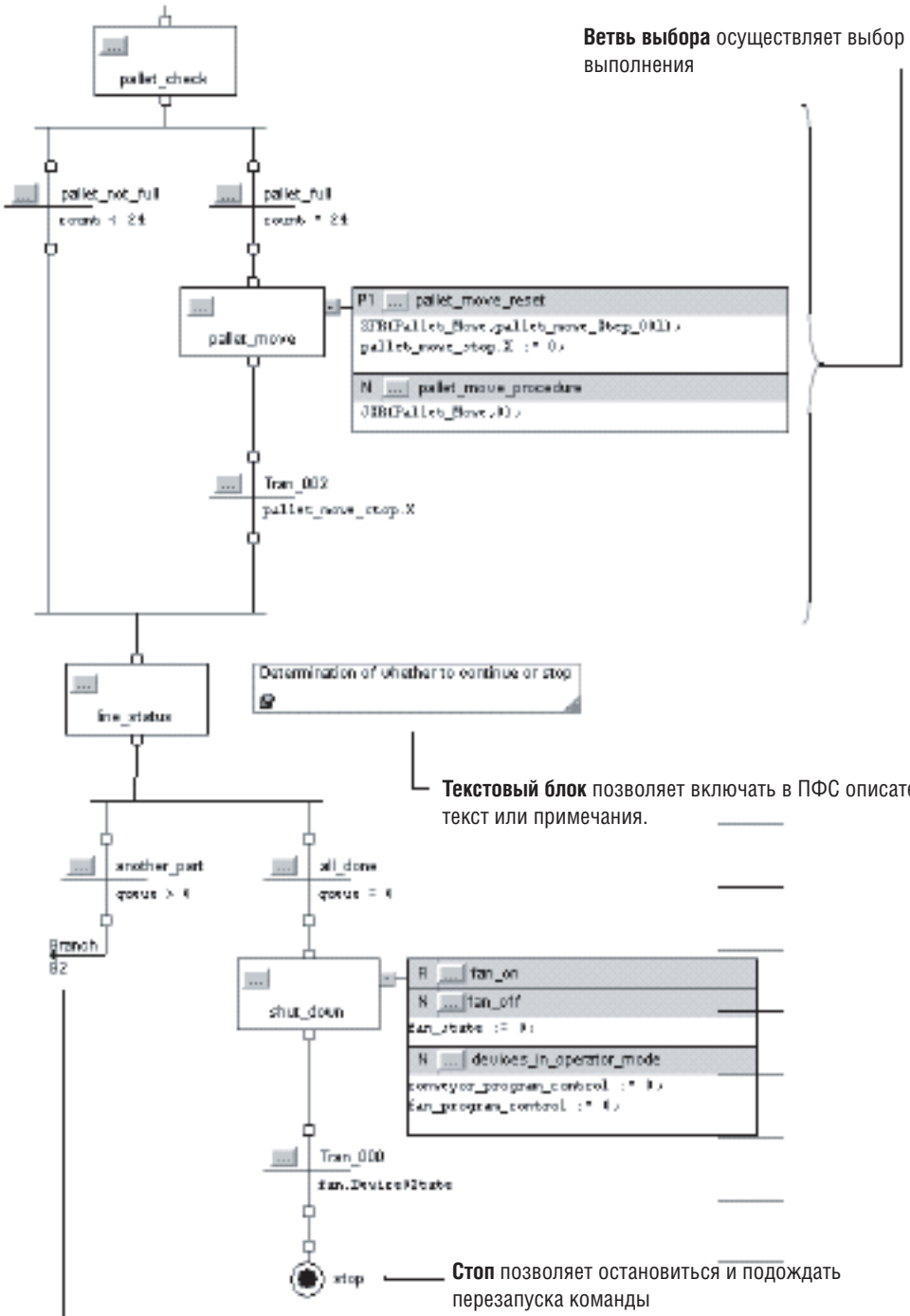
Пример ПФС, показывающий ее отдельные элементы, приводится на рисунках 5.1 и 5.2.

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:

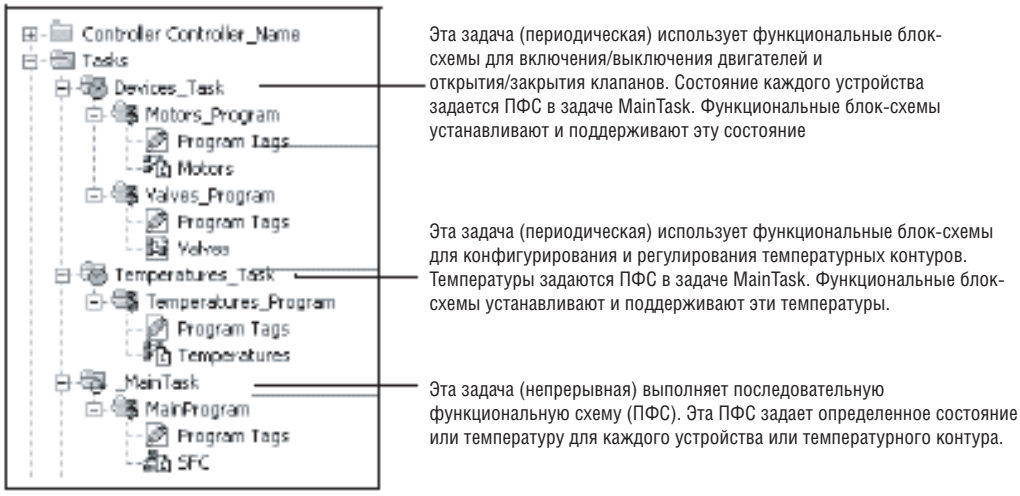
Эти функции:	Передаются:
конфигурирование и регулирование устройств	периодической задаче
управление переходом устройства в определенное состояние	ПФС в непрерывной задаче
установление последовательности выполнения вашего процесса	

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

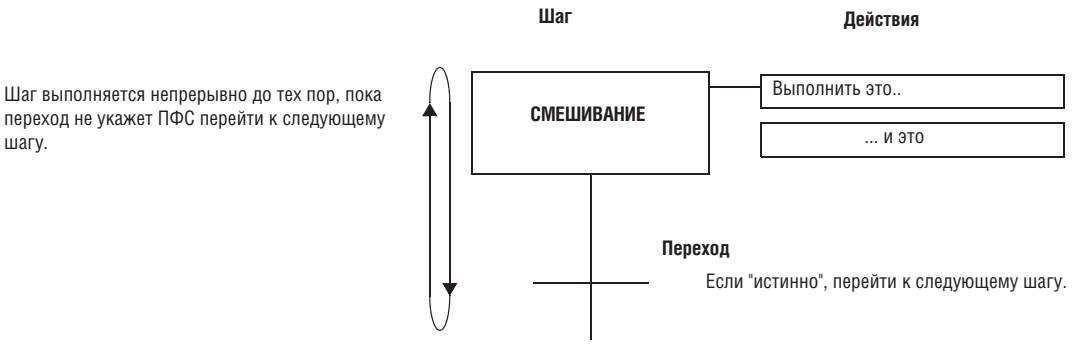
Для выполнения ПФС сконфигурируйте ее в качестве главной процедуры для программы или вызывайте ее как подпрограмму.

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.

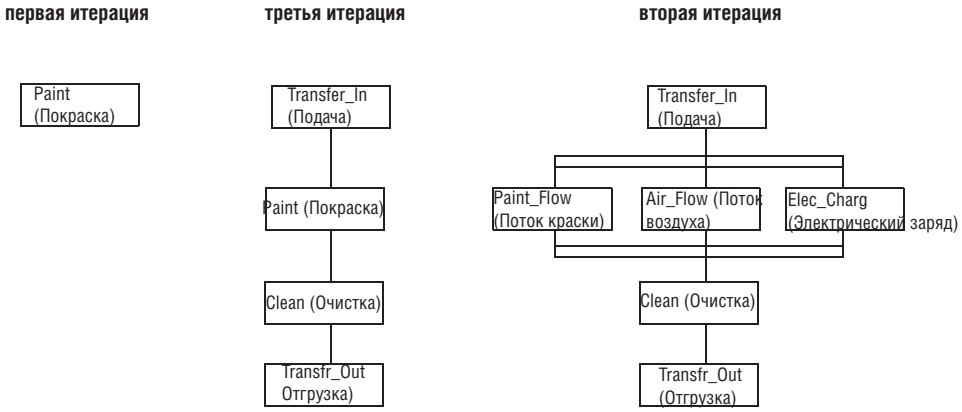


Переход заканчивает шаг. Переход определяет физические условия, которые должны иметь место или измениться, чтобы произошел переход к следующему шагу.

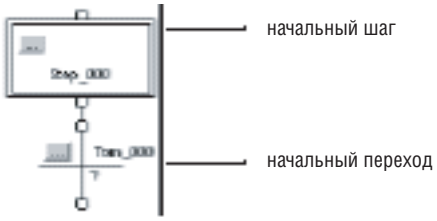
Руководящие указания

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):

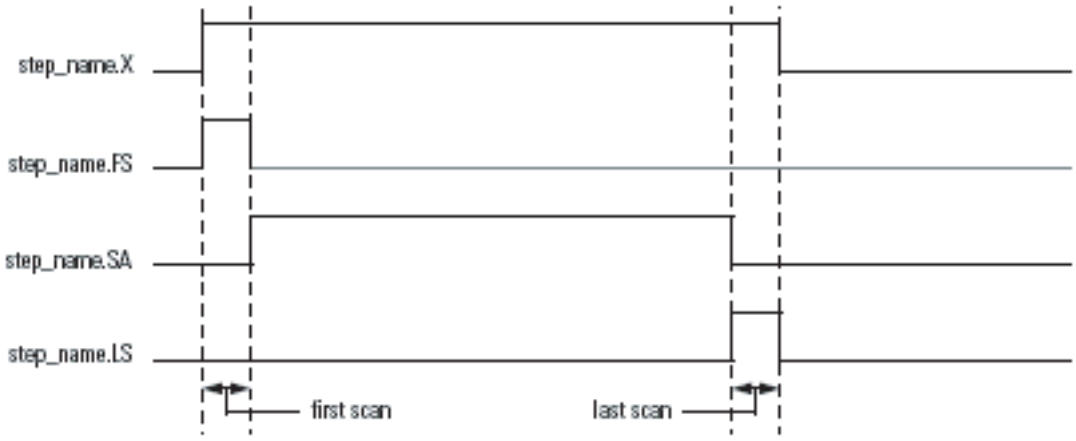
Если вы хотите:	То проверьте или установите этот член:	Тип данных:	Описание:
определить, сколько времени шаг был активен (в миллисекундах)	T	DINT	Когда шаг становится активным, значение Timer (T) сбрасывается и начинается прямой отсчет в миллисекундах. Таймер продолжает отсчет, пока данный шаг не станет 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

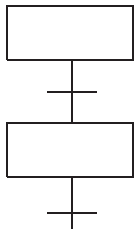
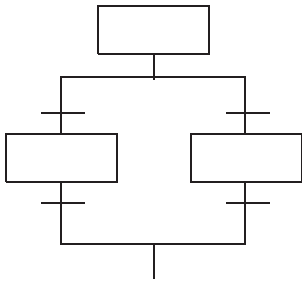
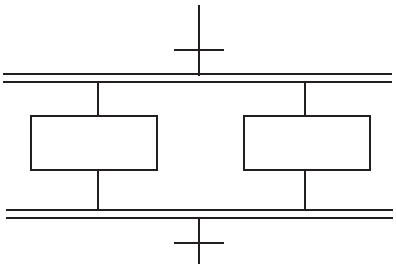
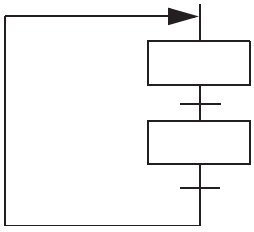
На следующей схема показана взаимосвязь битов X, FS, SA и LS.


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

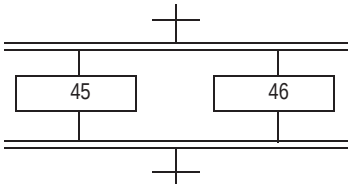
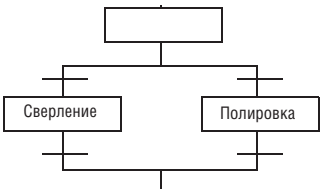
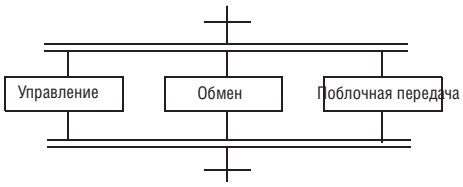
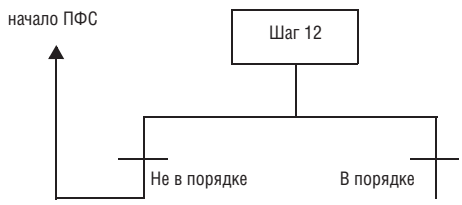
Организация шагов

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

Обзор

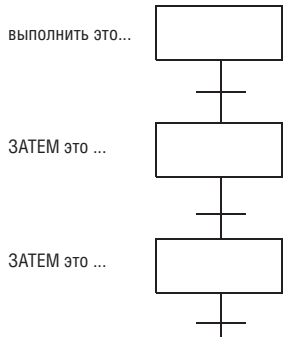
Для:	Используйте такую структуру:	С учетом следующего:
Последовательного выполнения одного или нескольких шагов: • Циклически выполняется один шаг. • Затем циклически выполняется следующий шаг.	**Последовательность** 	ПФС проверяет переход в конце шага: • Если «истина», ПФС переходит к следующему шагу. • 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.



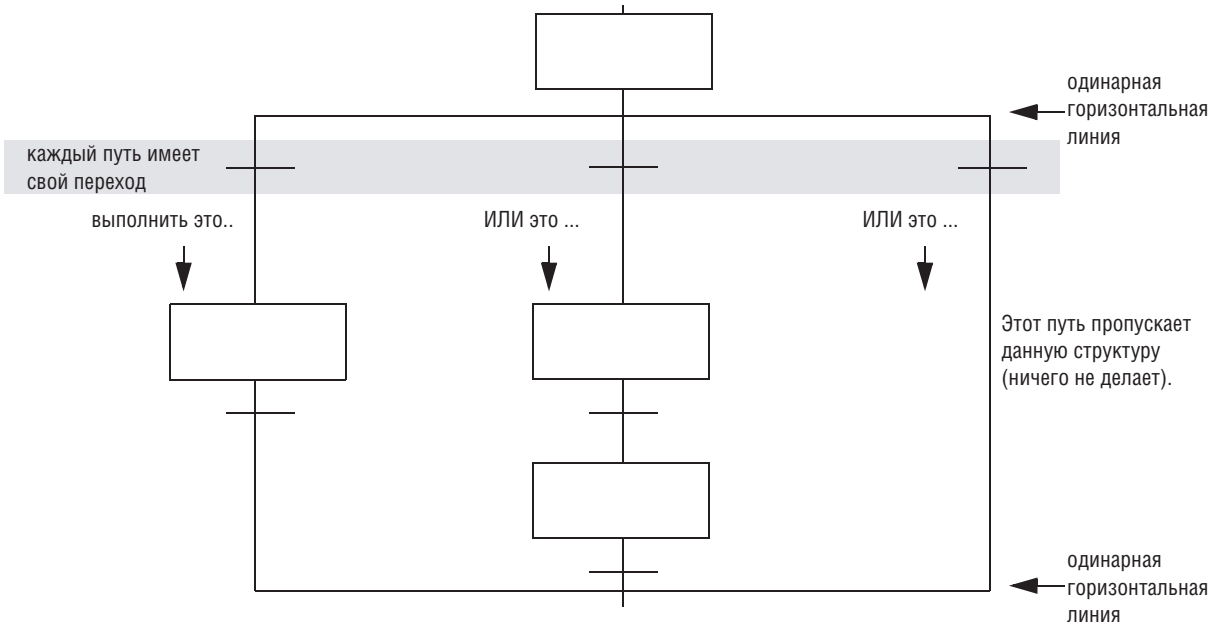
Подробная схема выполнения последовательности шагов приводится на рис. 5.5 на стр. 5-52.

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.



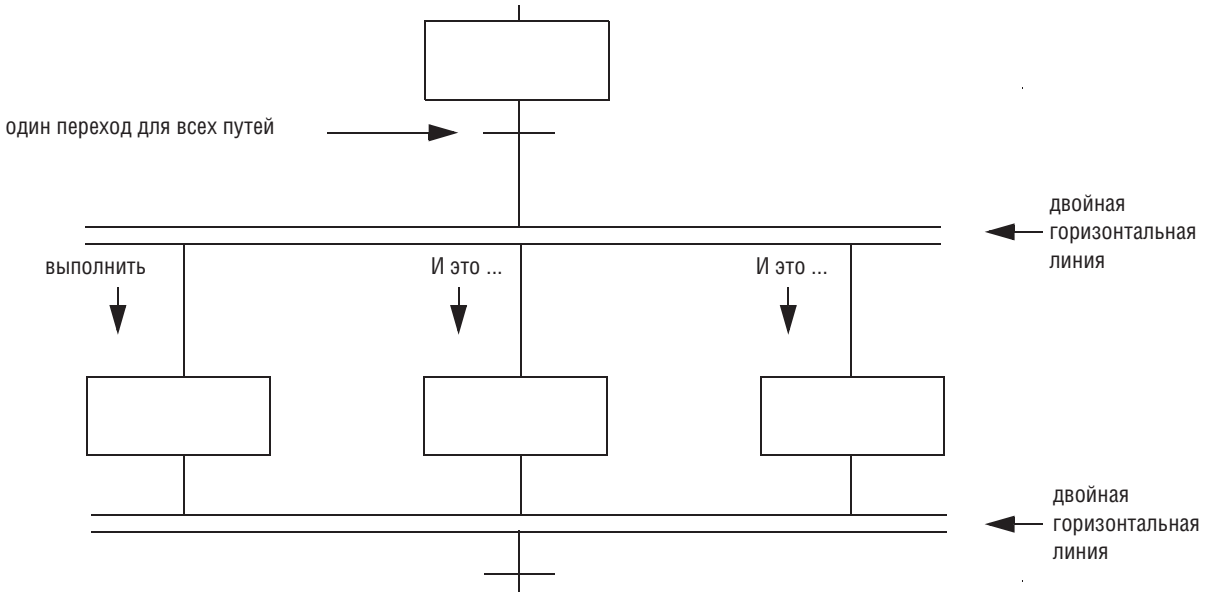
Подробная схема выполнения ветви выбора приводится на рис. 5.7 на стр. 5-54.

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.

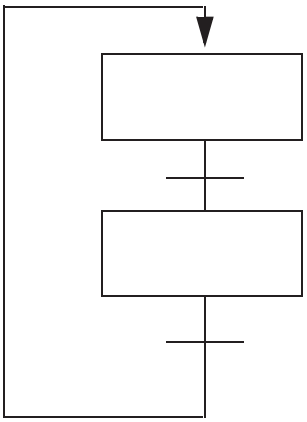
Информацию о том, как обойти ветвь и воспрепятствовать выполнению пути, вы найдете в разделе «Форсировка логических элементов» на стр. 14-1.

Связь, ведущая к предыдущему шагу

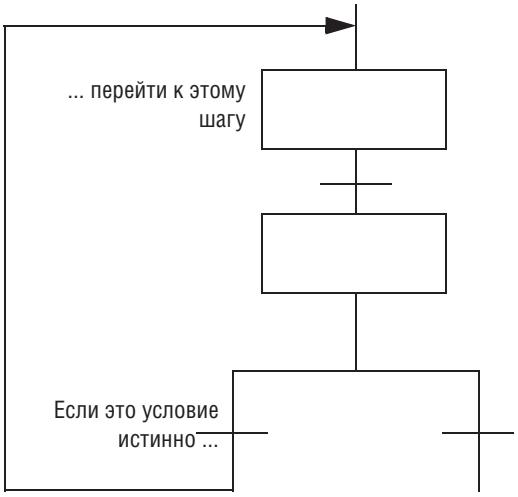
Помимо соединения шагов в виде последовательностей, одновременных ветвей и ветвей выбора вы можете соединить шаг с предыдущей точкой вашей ПФС. Это позволяет вам:

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

Например:



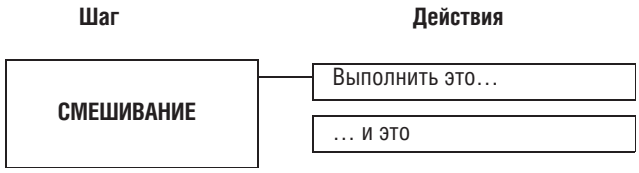
простой цикл, повторяющий выполнение всей ПФС



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

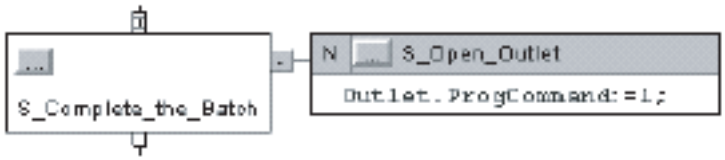
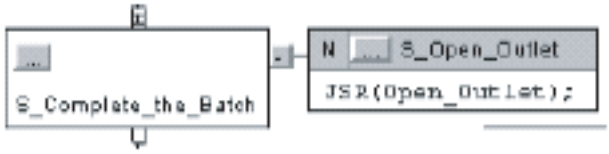
Небулево действие содержит логику для этого действия. Оно использует структурированный текст для выполнения присваиваний и инструкций или вызова подпрограммы.

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.

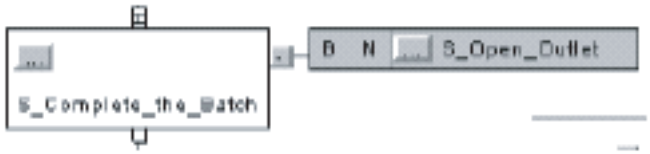
Небулево действие *нельзя* повторно использовать внутри одной и той же ПФС, за исключением его использования для сброса хранимого действия. На одну ПФС допускается лишь один экземпляр каждого небулева действия.

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:

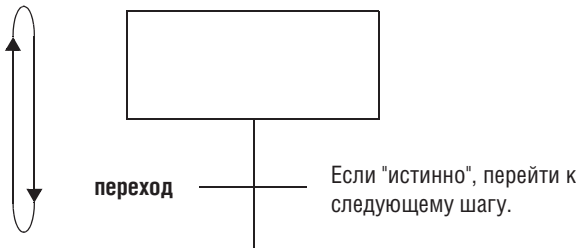
Таблица 5.1 Выбор определителя для действия

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)

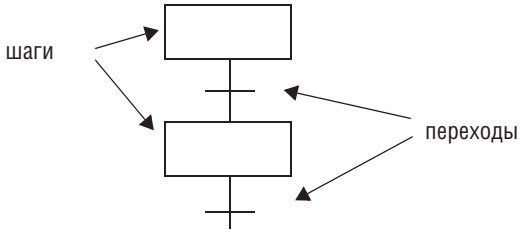
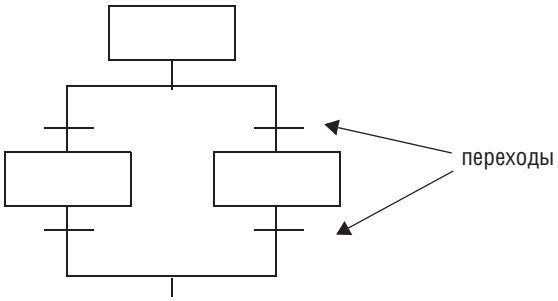
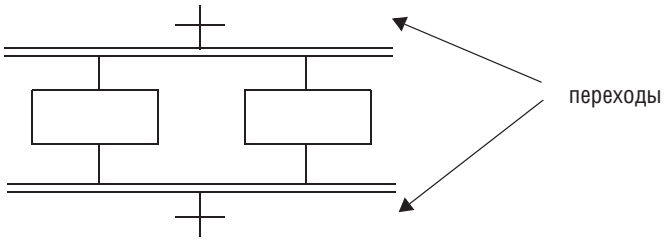
Задание условий перехода

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:

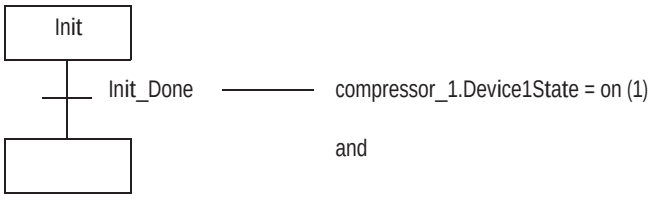
ПРИМЕР

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:



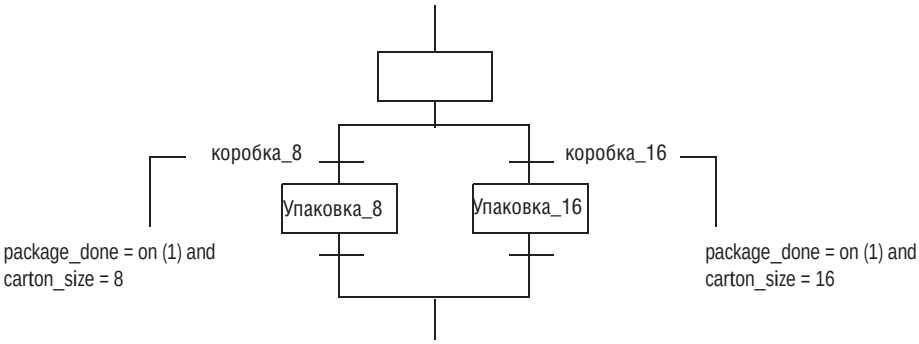
ПРИМЕР

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?

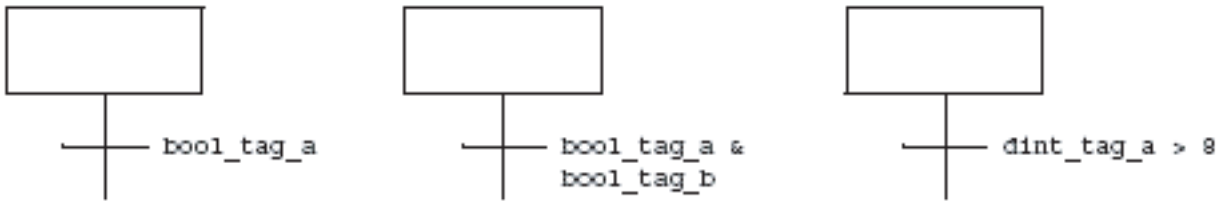
Вы можете запрограммировать переход следующими способами:

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

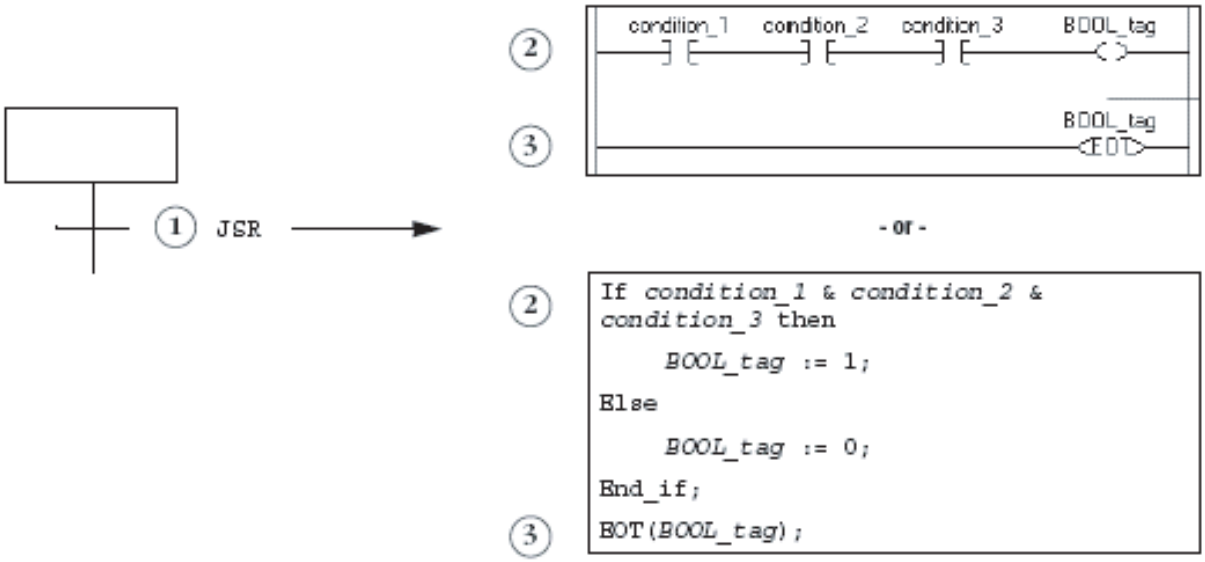
Самым простым способом программирования перехода является ввод условий в структурированный текст в виде **выражения BOOL.** Выражение BOOL использует булевы теги, операторы отношения и логические операторы для сравнения значений или проверки условий на истинность или ложность. Например, `tag1>65`.

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

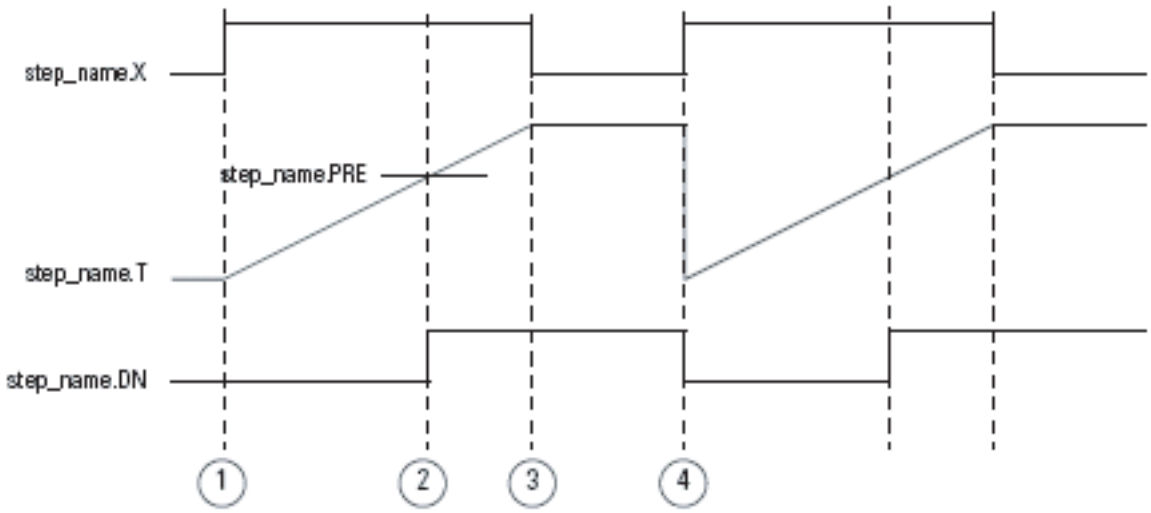
3. Используйте инструкцию EOT для того, чтобы установить состояние перехода равным значению тега типа BOOL. Когда этот тег установлен (истинен), переход является истинным.

Переход по истечении заданного времени

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

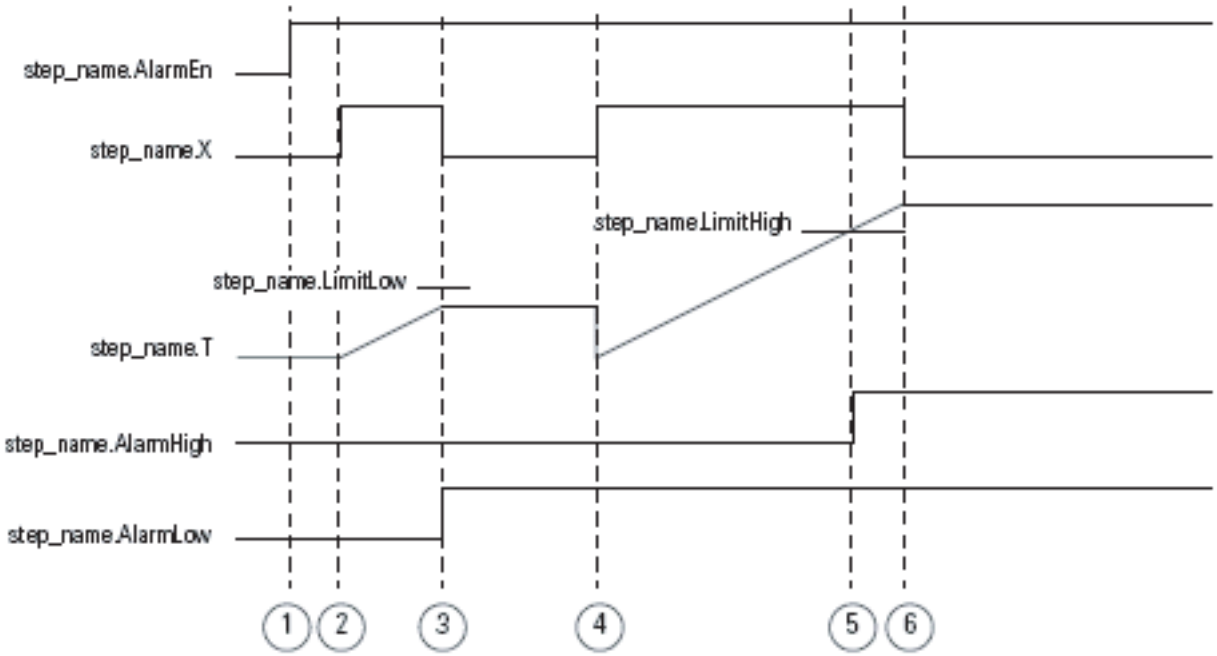
- сигнализации того, что данный шаг проработал требуемое время и ПФС надлежит перейти к следующему шагу
- сигнализации того, что шаг выполняется слишком долго и ПФС надлежит перейти к шагу ошибки

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.

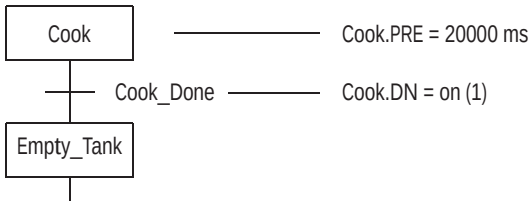
ПРИМЕР

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.

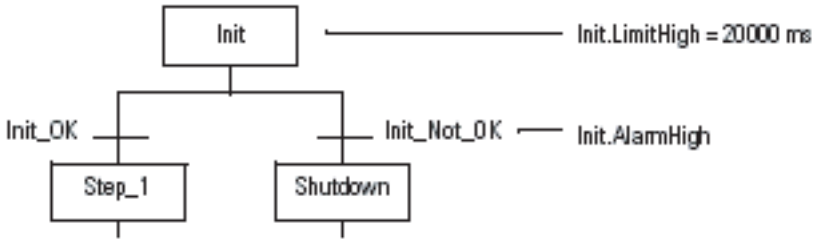
ПРИМЕР

В функциональной спецификации указывается:

а. Привести в исходное состояние 8 устройств.

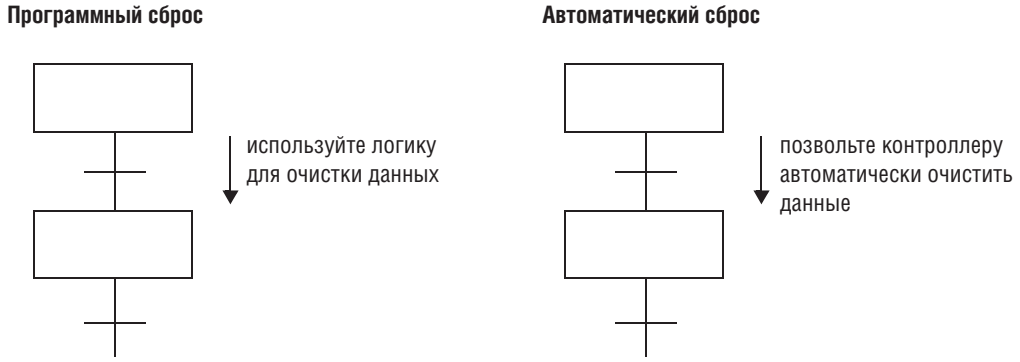
б. Если все 8 устройств не вернутся в исходное состояние в течение 20 секунд, то остановить систему.

Решение:



Выключение устройства в конце шага

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



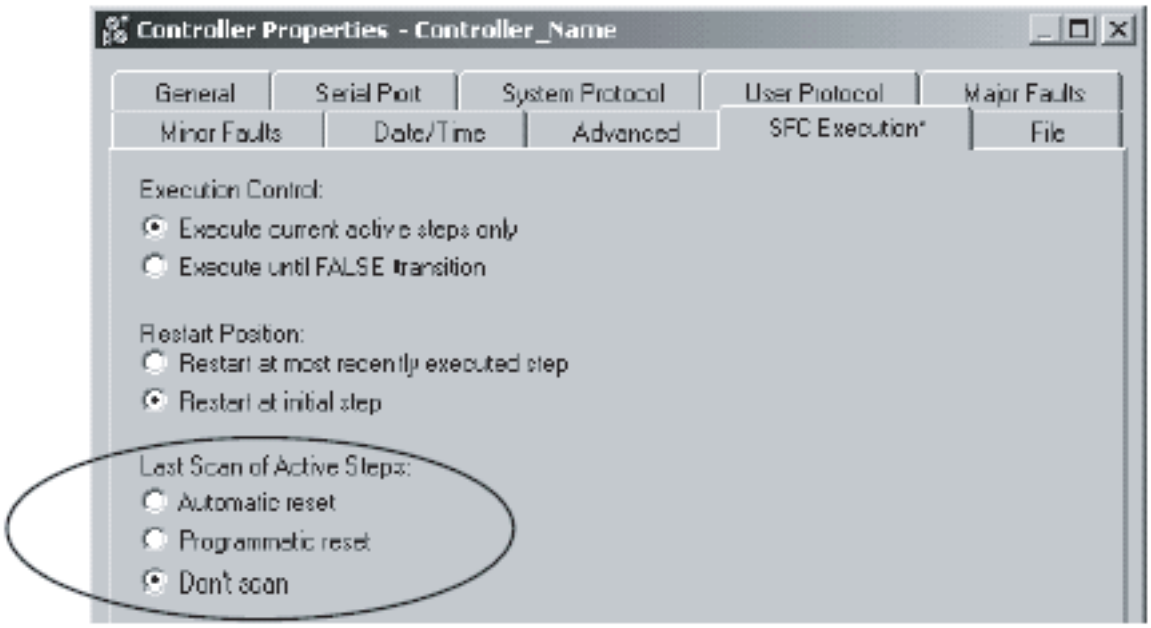
Каждый из этих вариантов требует от вас:

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).

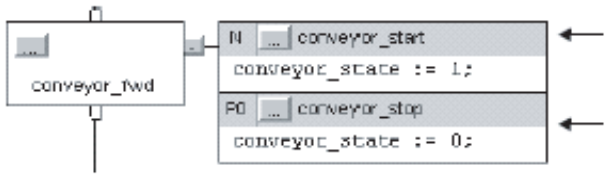
2.. Используйте действие *P0 Pulse (Falling Edge)* (Задний фронт импульса) для очистки соответствующих данных. Убедитесь в том, что действие или действия P0 являются последними в последовательности действий для данного шага.

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:

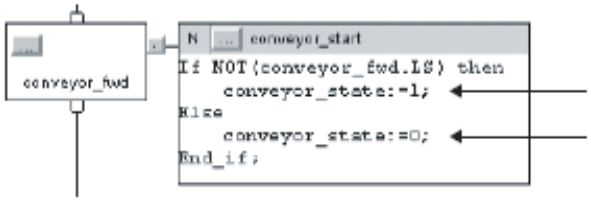
- К вашей обычной логике добавьте логику, очищающую соответствующие данные. Используйте бит LS данного шага или бит Q соответствующего действия для обусловливания выполнения логики.
- Используйте действие *P0 Pulse (Falling Edge)* (Задний фронт импульса)для очистки соответствующих данных. Убедитесь в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

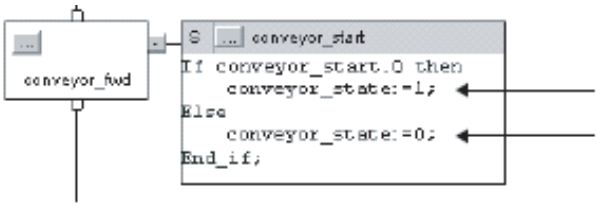


Когда шаг находится не на последнем сканировании (*conveyor_fwd.LS* = 0), этот оператор устанавливает *conveyor_state*. Когда *conveyor_state* устанавливается, конвейер включается.

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

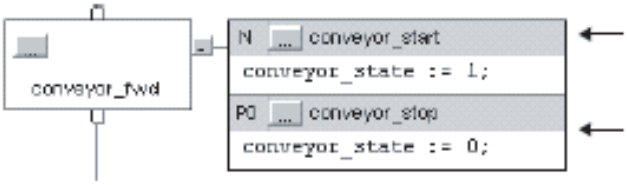


Когда шаг находится не на последнем сканировании (*conveyor_start.Q* = 1), этот оператор устанавливает *conveyor_state*. Когда *conveyor_state* устанавливается, конвейер включается.

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.

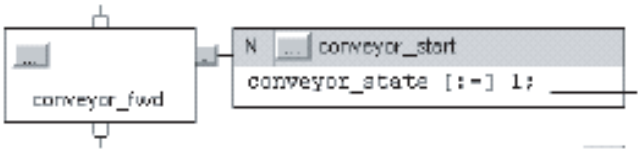
Как правило, пост-сканирование выполняет инструкции так, как если бы все условия были ложными. Например, инструкция Output Energize (OTE) в процессе пост-сканирования очищает свои данные.

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

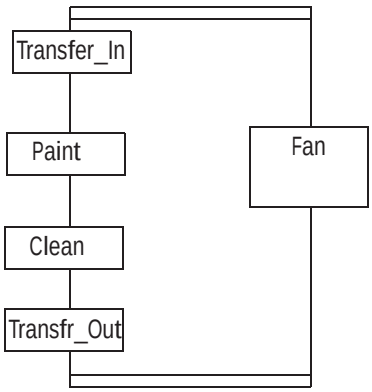
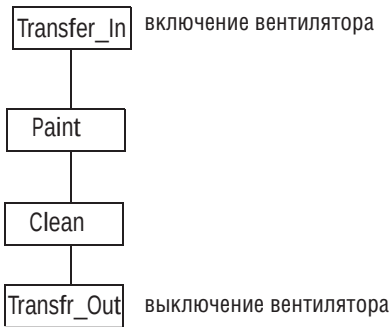
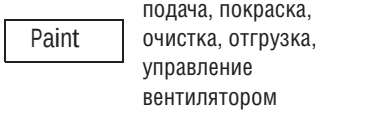


Это действие включает конвейер. Когда *conveyor_state* устанавливается, конвейер включается.

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.

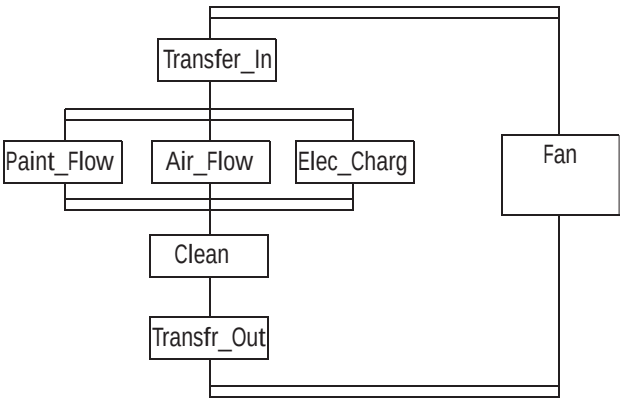
Далее приводится пример использования такого способа.

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:

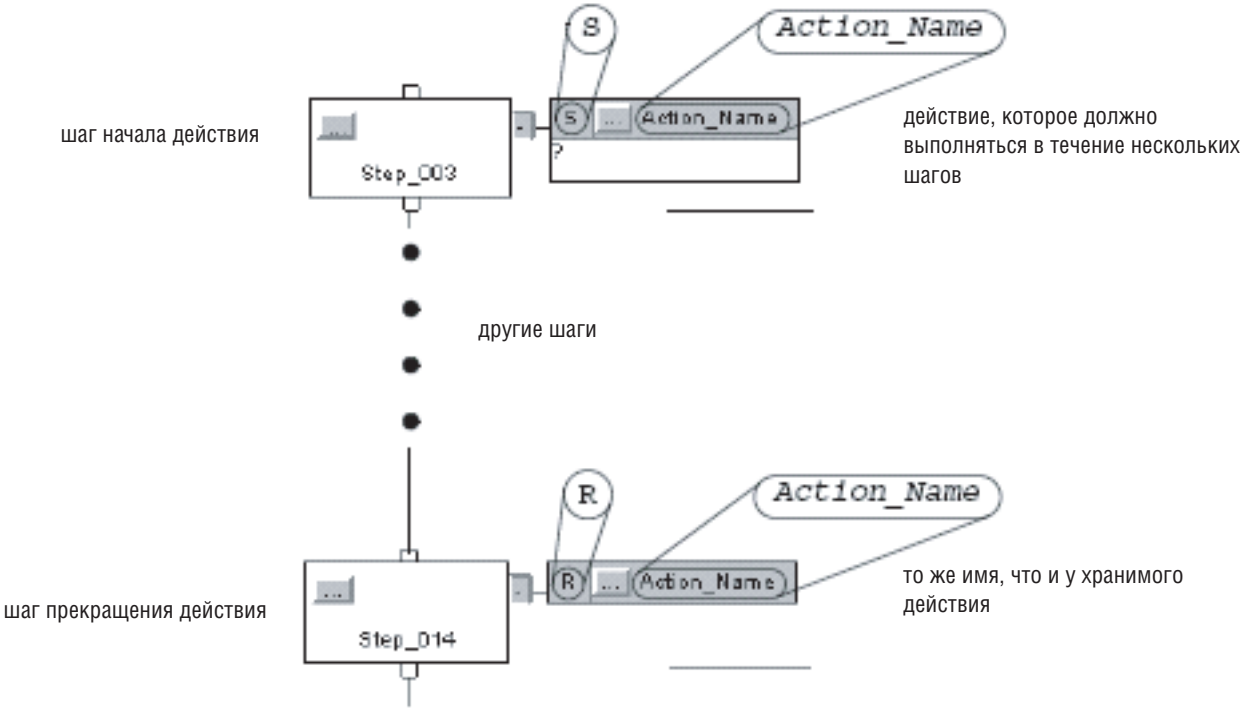


Сохранение и сброс действия

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.

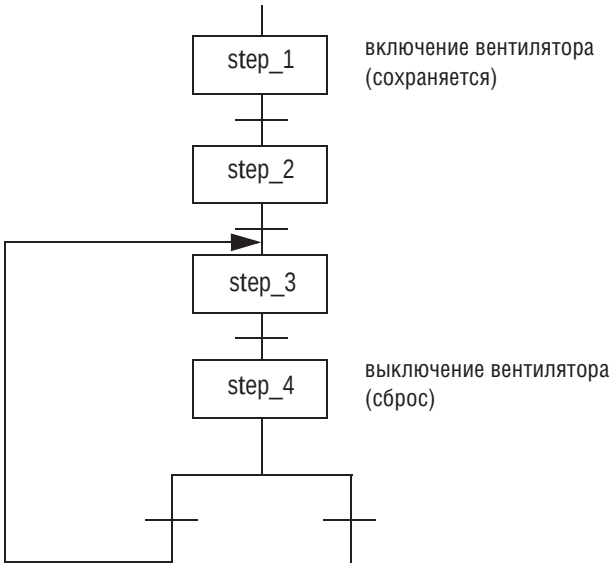


Когда ПФС выходит из шага, где хранится данное действие, программное обеспечение RSLogix 5000 продолжает показывать хранимое действие как активное. (По умолчанию это действие заключено в зеленую рамку.) Это информирует вас о том, что ПФС выполняет логику данного действия.

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.

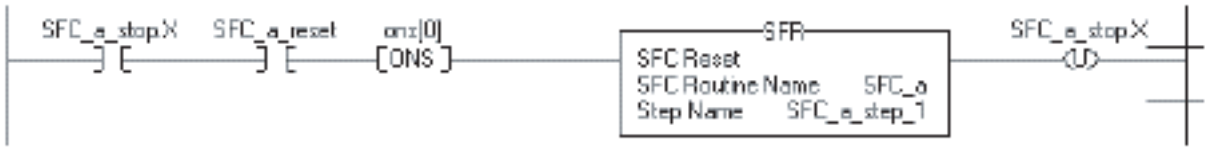
ПРИМЕР

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:

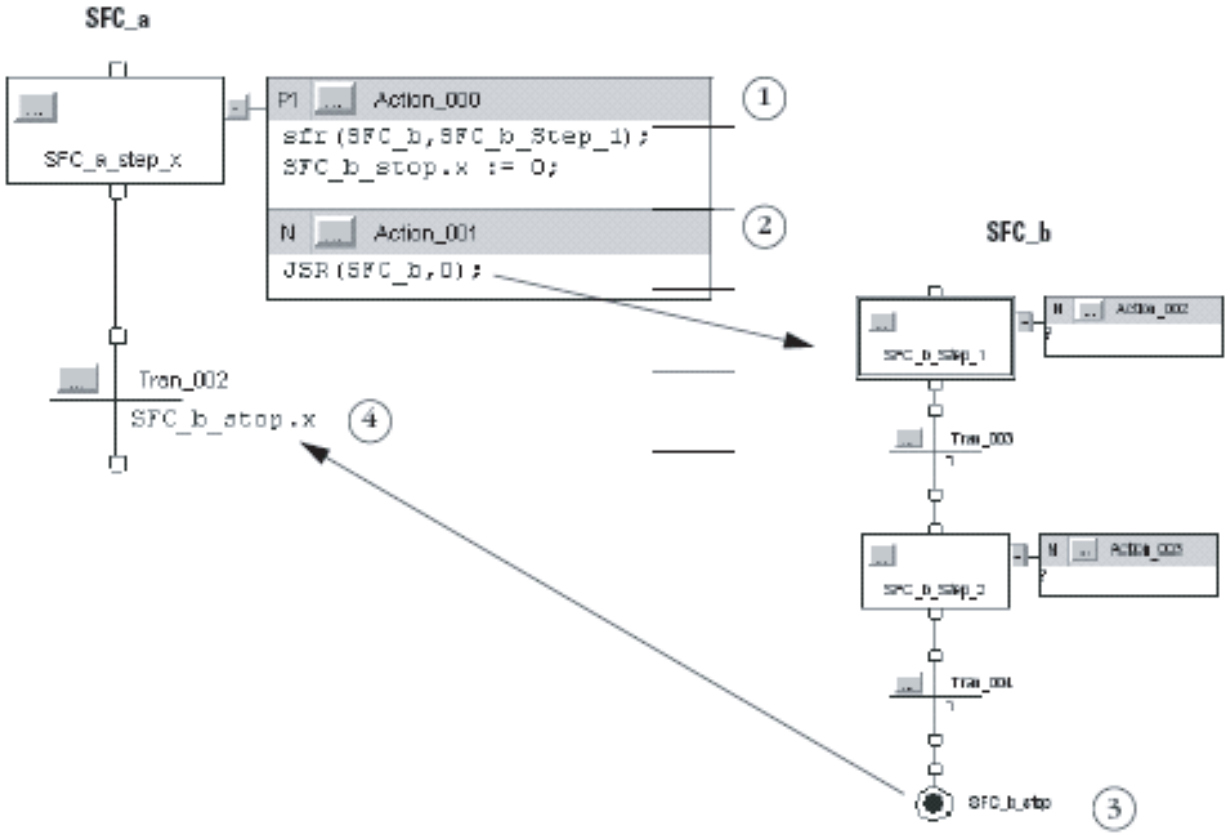
Если вы хотите:	То проверьте или установите этот член:	Тип данных:	Описание:
определить, когда ПФС дойдет до стопа	X	BOOL	• Когда ПФС доходит до стопа, устанавливается бит X. • Бит X обнулится при переходе контроллера из программного режима в режим выполнения, если вы сконфигурировали ПФС на перезапуск 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.

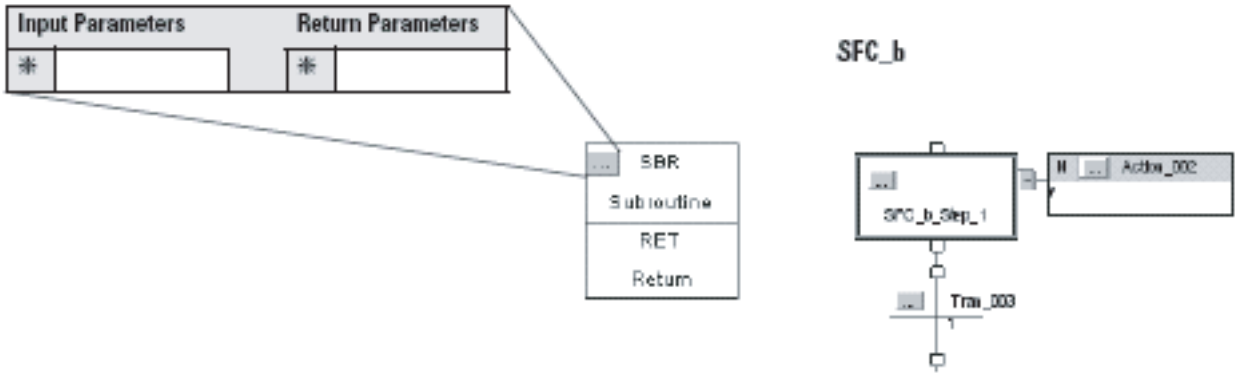
2. Вызовите ПФС *SFC_b*.

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.

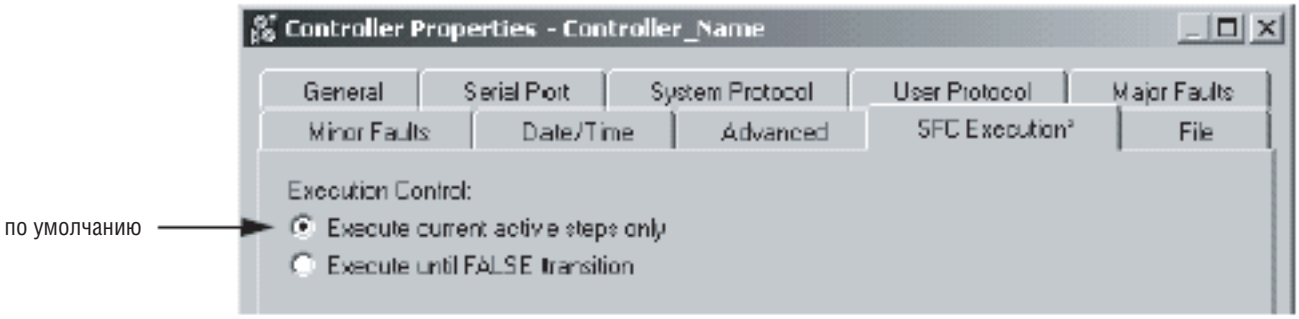
Передача параметров

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.

Следующая схема:	Приводится на стр.:
Выполнение последовательности	5-52
Выполнение одновременной ветви	5-53
Выполнение ветви выбора	5-54
Вход/выход параметров в/из ПФС	5-54
Варианты управления выполнением	5-55

Рисунок 5.5 Выполнение последовательности

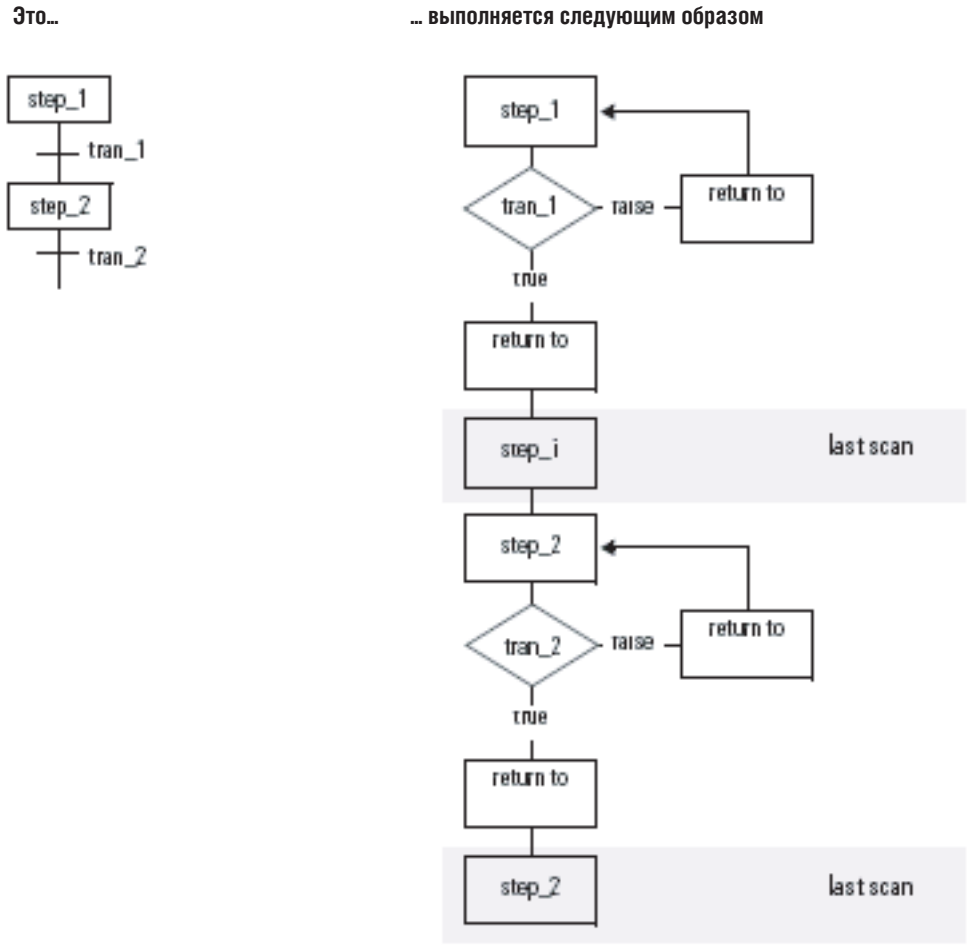
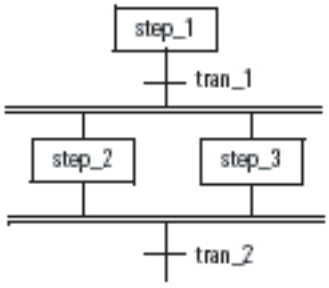


Рисунок 5.6 Выполнение одновременной ветви

Это...



... выполняется следующим образом

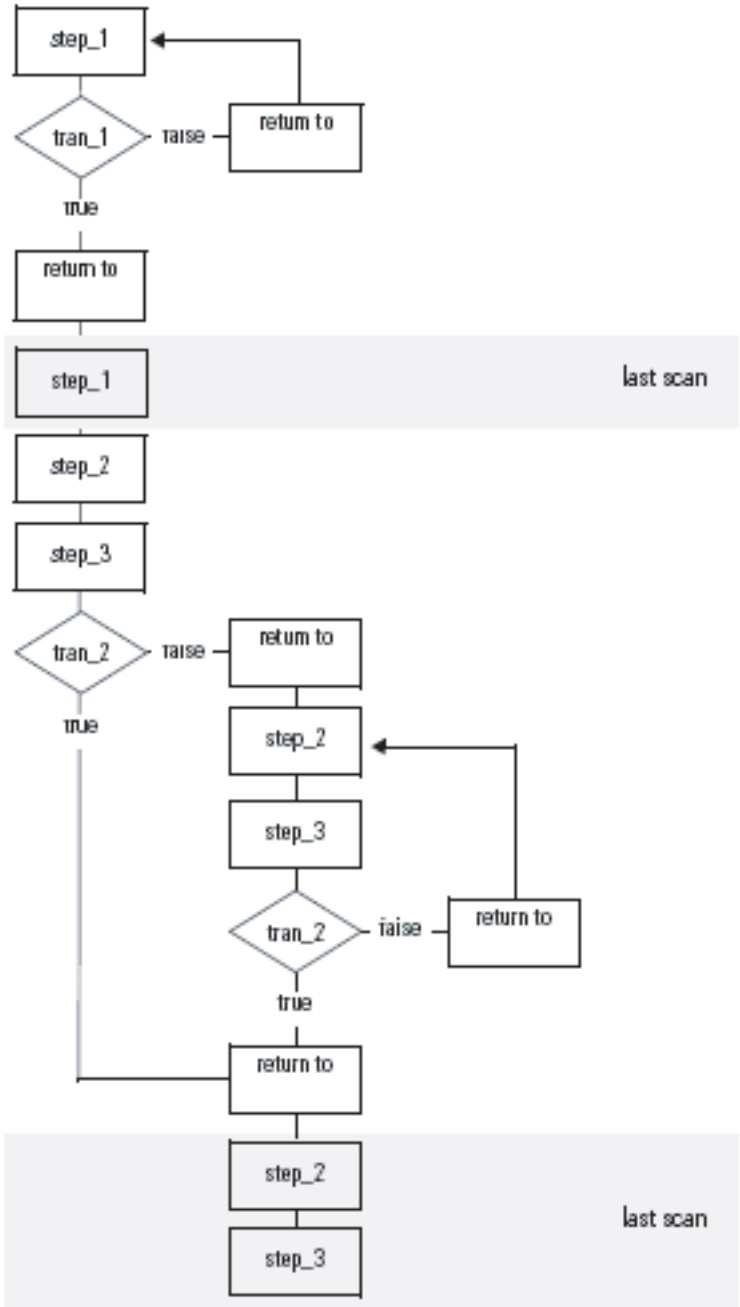


Рисунок 5.7 Выполнение ветви выбора

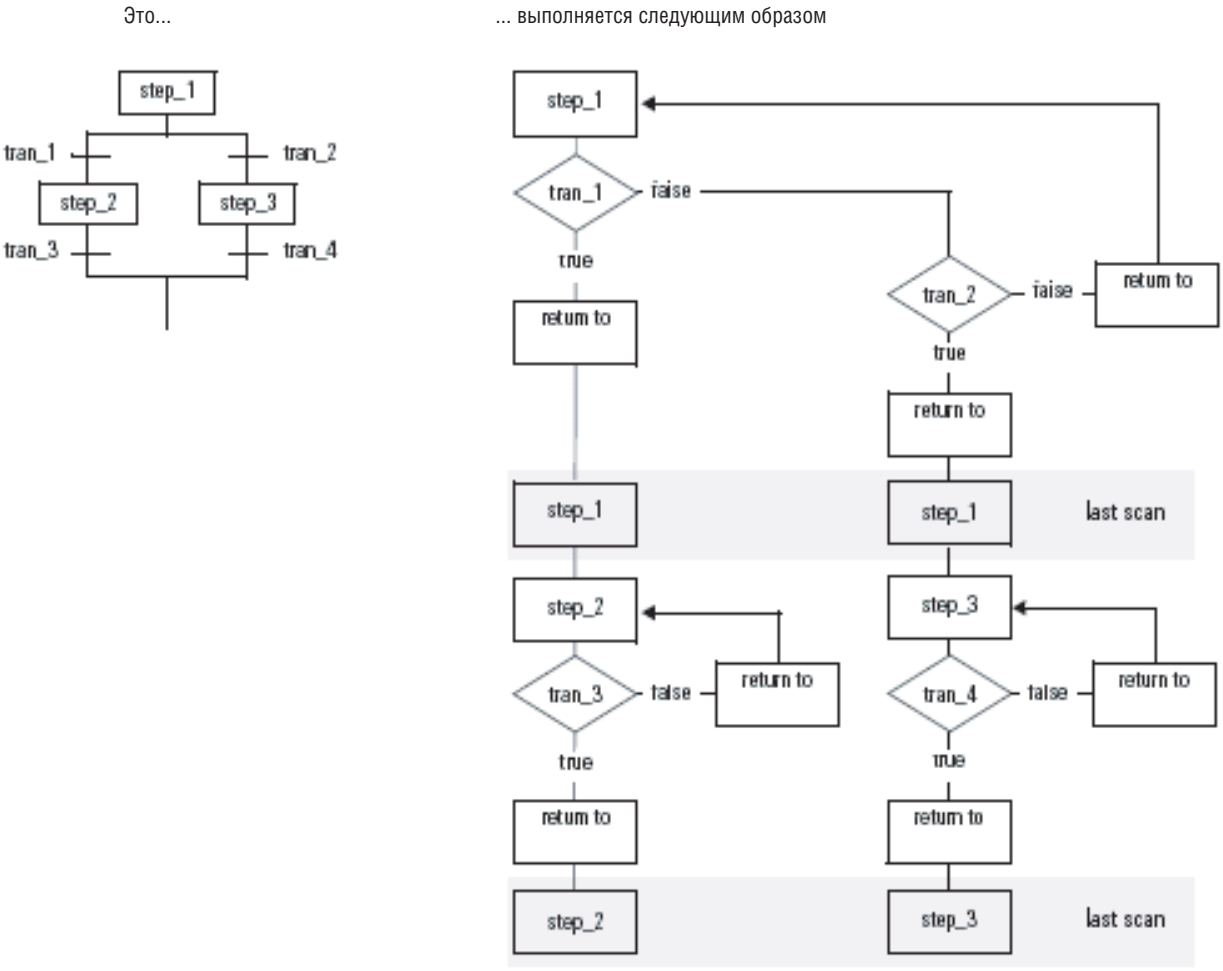


Рисунок 5.8 Вход/выход параметров в/из ПФС

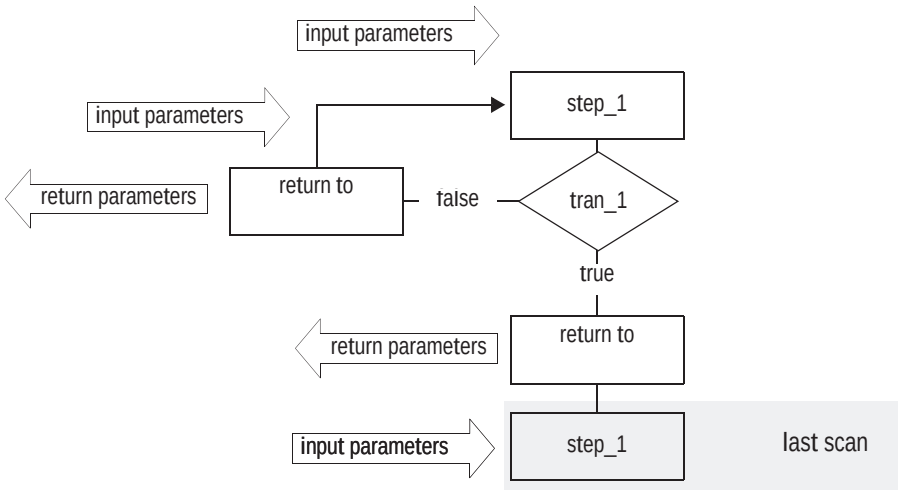
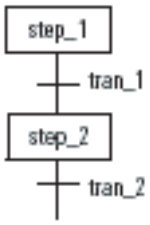


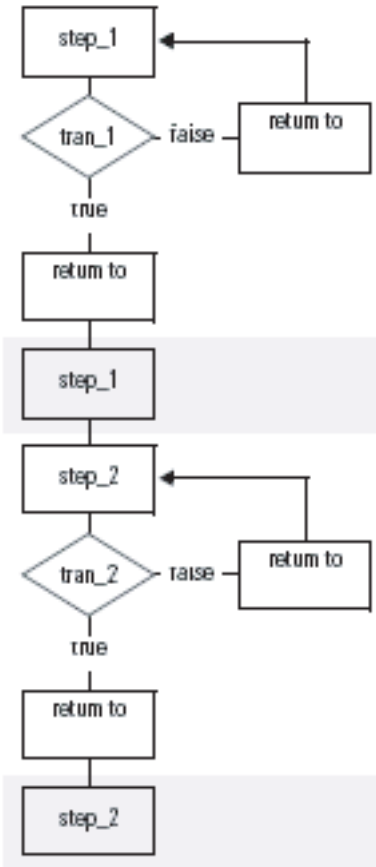
Рисунок 5.9 Варианты управления выполнением

Это...

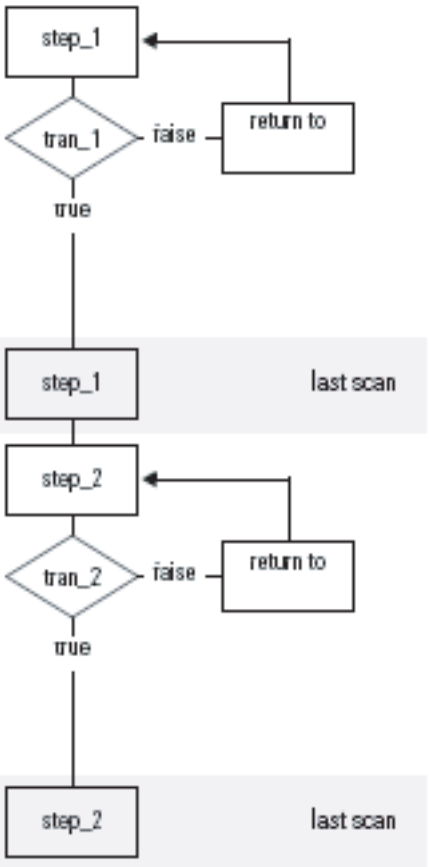


... выполняется следующим образом

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.

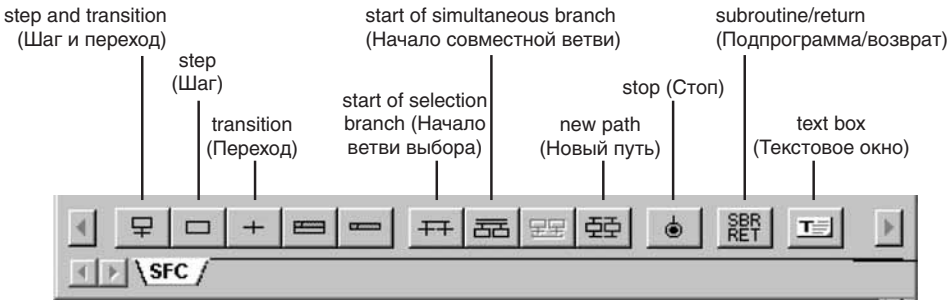
Как использовать данную процедуру

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.



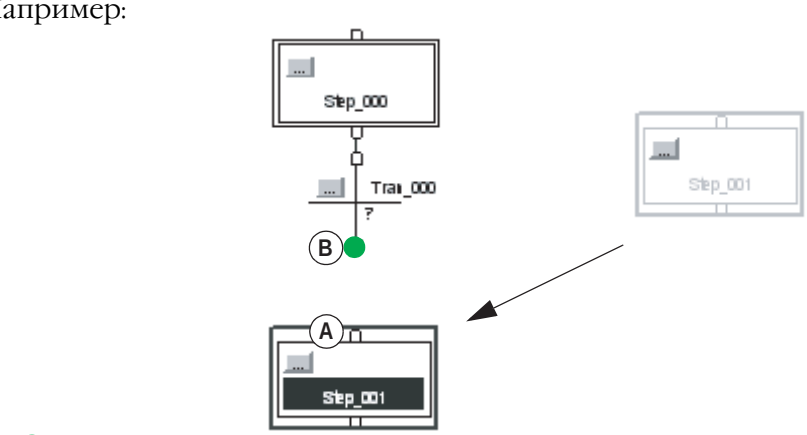
Для того, чтобы добавить элемент к вашей ПФС, у вас имеются следующие опции:

- ☐ 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:



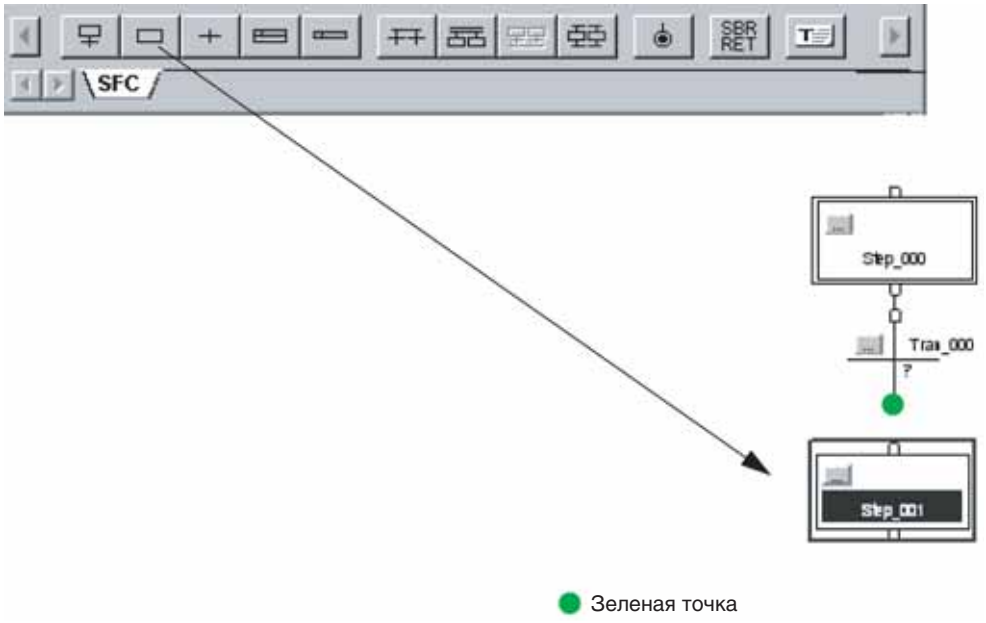
3. Соедините (скоммутируйте) два элемента друг с другом, щелкните на контакте элемента (A) , а затем на контакте другого элемента (B) . Зеленая точка указывает подходящее место контакта.

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.

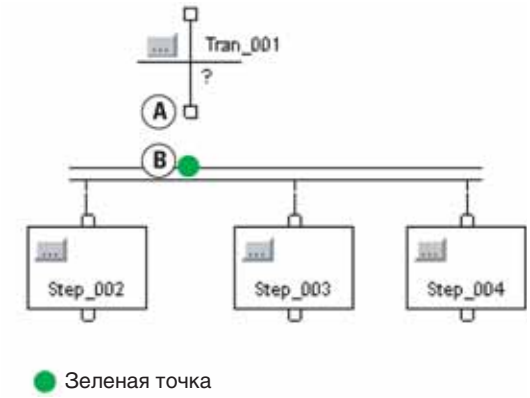


Создание совместной ветви

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.

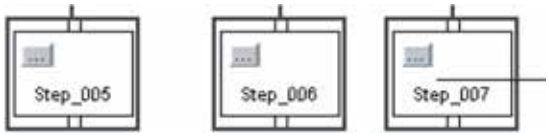
   Затем щелкните на кнопке. 



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.

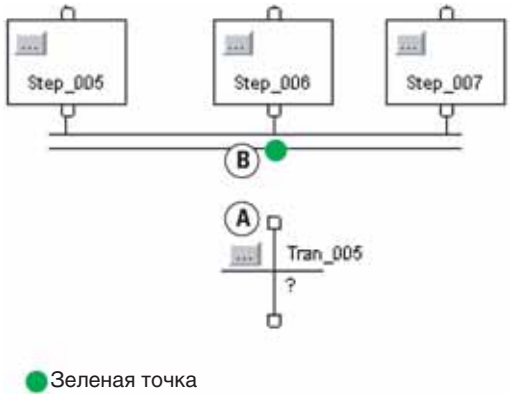
Окончание совместной ветви

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.

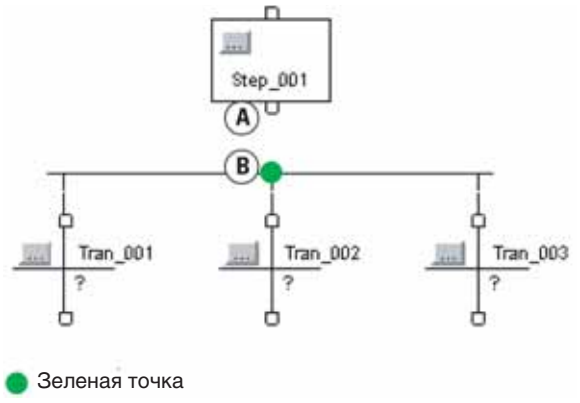


4. Чтобы подключить совместную ветвь к этому переходу, щелкните на верхнем контакте этого перехода (A) , а затем щелкните на горизонтальной линии этой ветви (B) . Зеленая точка указывает подходящее место контакта.

Создание ветви выбора

Начало ветви выбора

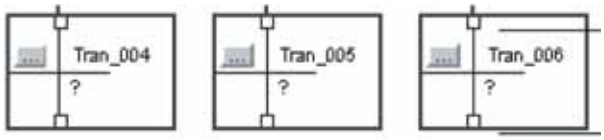
1. Щелкните на кнопке а на панели инструментов ПФС. Затем отбуксируйте эту новую ветвь в желаемое место.
2. Чтобы добавить путь к этой ветви, выберите (щелчком) первый переход пути, который находится слева от того места, где вы хотите добавить новый путь. Затем щелкните на кнопке а .



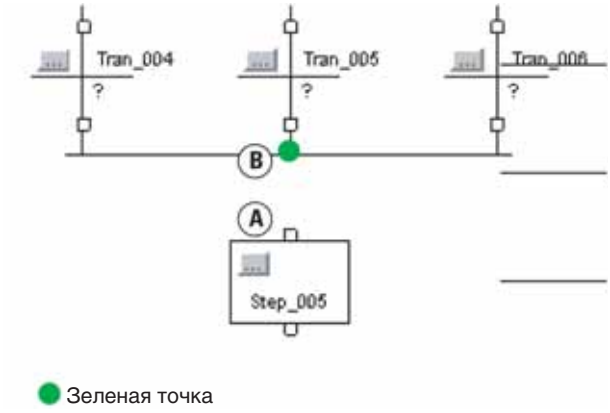
3. Чтобы подключить ветвь выбора к предыдущему шагу, щелкните на нижнем контакте этого шага (A) , а затем щелкните на горизонтальной линии ветви (B) . Зеленая точка указывает подходящее место контакта.

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.



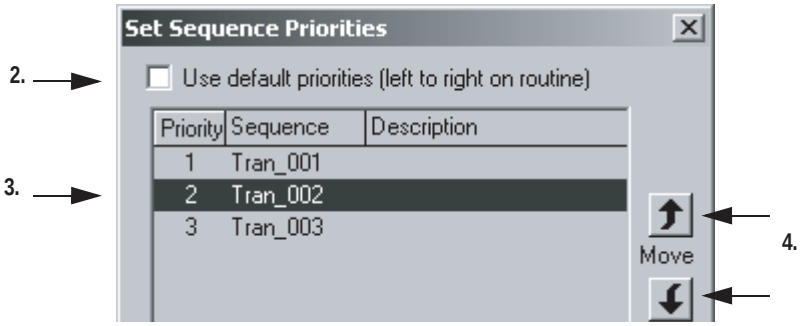
4. Чтобы подключить ветвь выбора к шагу, щелкните на верхнем контакте этого шага (A), а затем щелкните горизонтальной линии этой ветви (B). Зеленая точка указывает подходящее место контакта.

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.

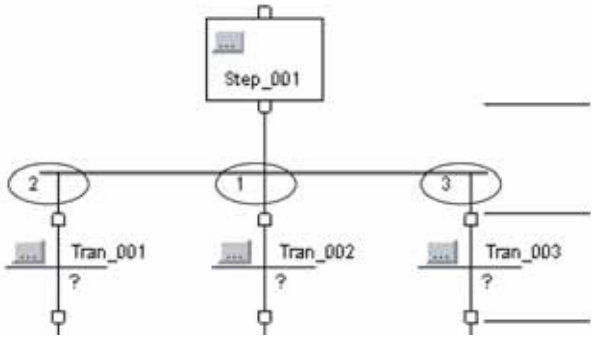
Чтобы присвоить приоритеты ветви выбора:

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).

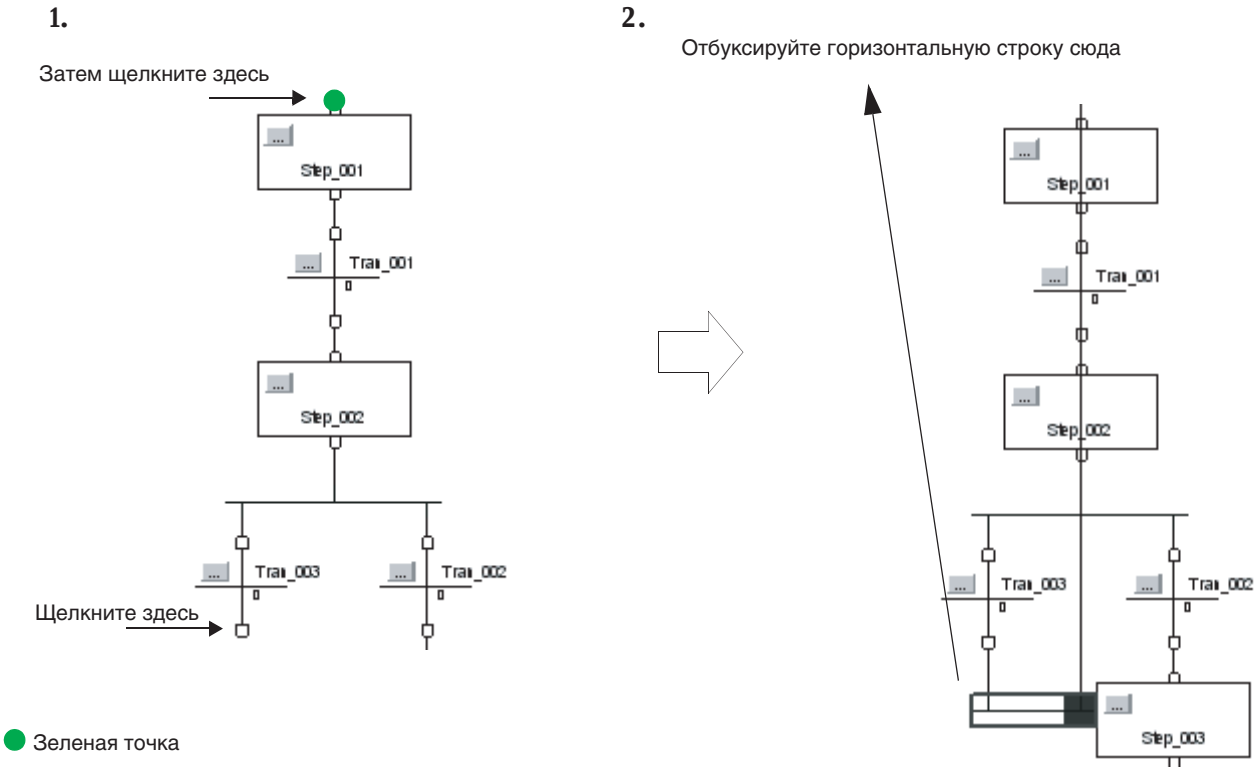
Подключение связи к шагу

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.

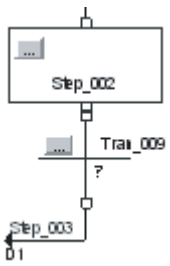
Например, переход к *Step_001* от *Tran_003*:



Скрытая связь

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

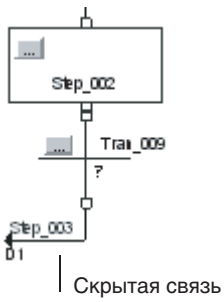
Чтобы спрятать связь, щелкните правой клавишей мыши на этой связи и выберите *Hide Wire (Спрятать связь)*.



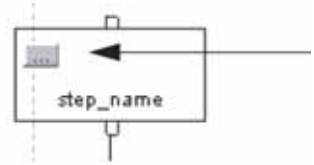
Чтобы увидеть элементы ПФС, к которым идет данная связь, щелкните в том месте, где располагается связь.

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)*.

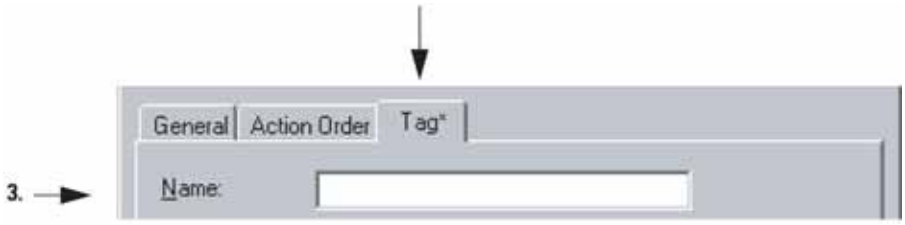


Изменение имени шага



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)*.



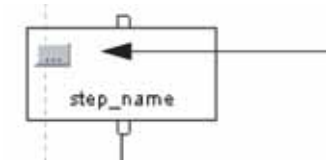
3. Ввести с клавиатуры новое имя для этого шага (тега).
4. Выбрать OK .

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.

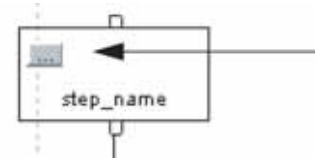
2. Введите с клавиатуры время для данного шага в миллисекундах.
3. Выберите OK.

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

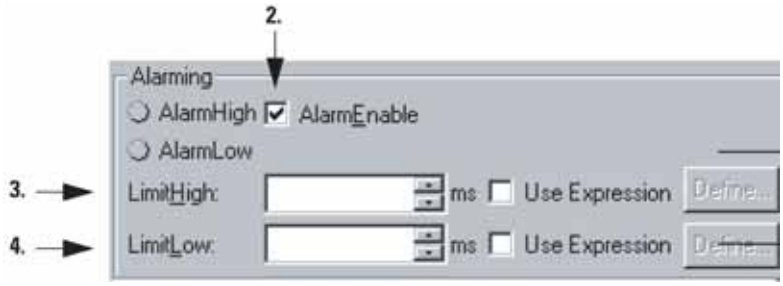
Для того чтобы узнать, как рассчитать заданное время для шага в режиме выполнения, обратитесь к разделу “Использование выражения для расчета времени» на стр. 6-12.

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 .

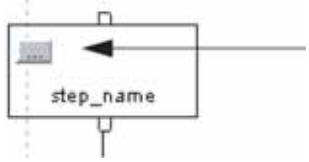
Для того чтобы узнать, как рассчитать заданное время для сигнала тревоги в режиме выполнения, обратитесь к разделу “Использование выражения для расчета времени» на стр. 6-12.

Использование выражения для расчета времени

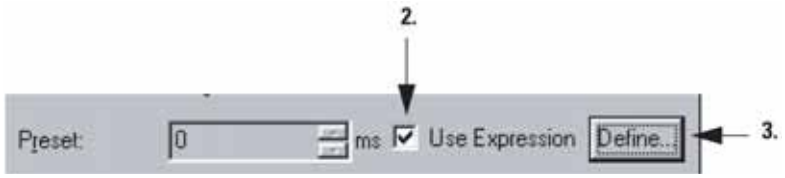
Для расчета времени на основе тегов в вашем проекте, введите числовое выражение. Вы можете использовать выражения для расчета следующих значений:

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

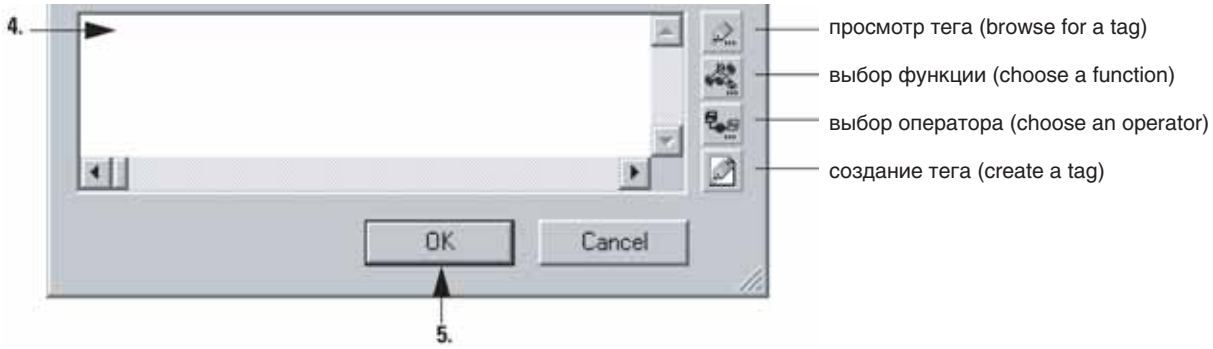
Чтобы ввести выражение для времени:



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



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



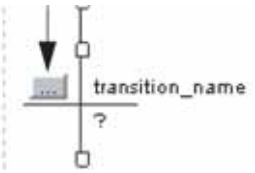
4. Введите с клавиатуры **числовое выражение**, которое определит данное время.

- 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)*.

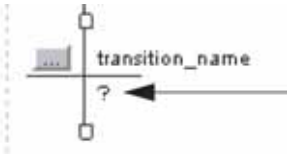
3. Введите с клавиатуры новое имя для данного перехода (тега).
4. Выберите [OK] .

Программирование перехода

Для программирования перехода вы можете использовать следующие опции:

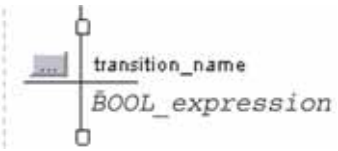
- Enter a BOOL Expression (Ввод булева выражения)
- Call a Subroutine (Вызов подпрограммы)

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.

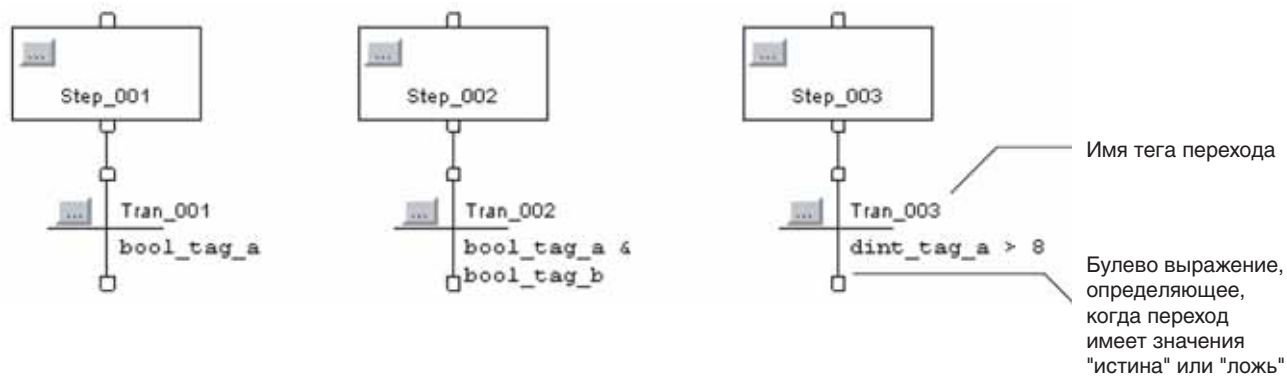
1. Дважды щелкните в текстовой области данного перехода.
2. Введите с клавиатуры булево выражение, определяющее, когда переход имеет значения «истина» или «ложь».
3. Чтобы закрыть окно текстового ввода, нажмите [Ctrl] + [Enter].

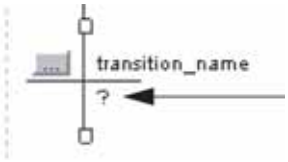


В примере, следующем ниже, представлены три перехода, использующих булевы выражения.

.

ПРИМЕР

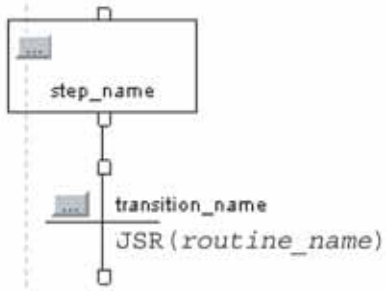




Вызов подпрограммы

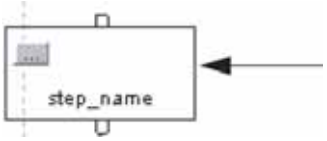
1. Щелкните правой клавишей мыши на переходе и выберите *Set JSR (Настройка JSR)*.

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

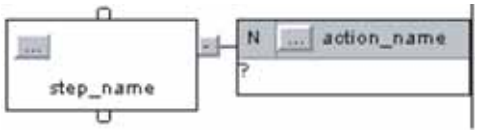


Добавление операции

Чтобы в рамках шага добавить операцию:

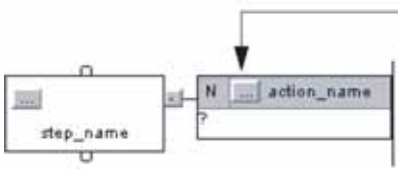


Щелкните правой клавишей мыши на шаге, в котором выполняется операция, и выберите *Add Action (Добавить операцию)*.



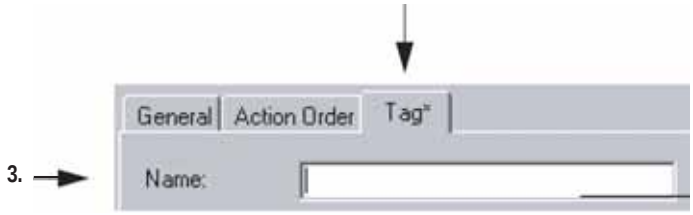
Переименование операции

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] .

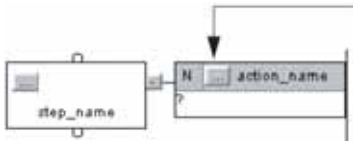
Конфигурирование операции

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.

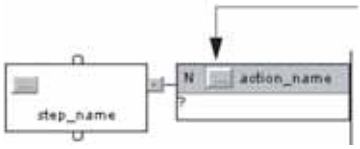
3. Если вы выбрали временной параметр, введите с клавиатуры временную границу или задержку для данной операции в миллисекундах. Параметры, связанные со временем, включают в себя:

- 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)

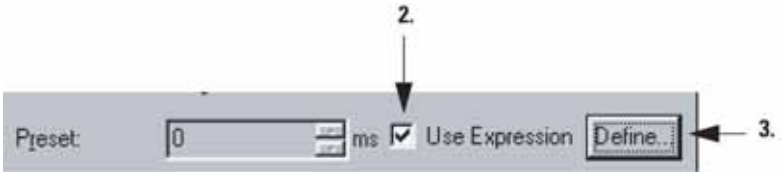
4. Выберите [OK] .

Расчет заданного времени при выполнении

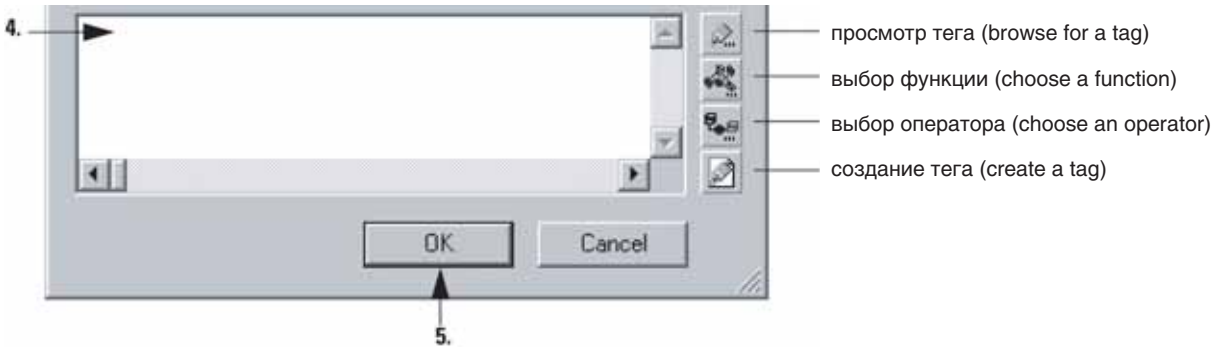
Для расчета заданного времени на основе тегов в вашем проекте, введите значение в виде **числового выражения**.



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



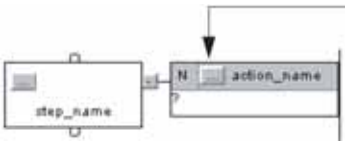
3. Щелкните на кнопке *Define (Определить)*.



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] .

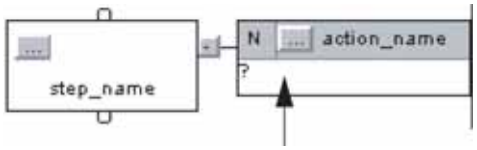
Программирование операции

Для программирования операции вы можете использовать следующие опции:

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

Ввод структурированного текста

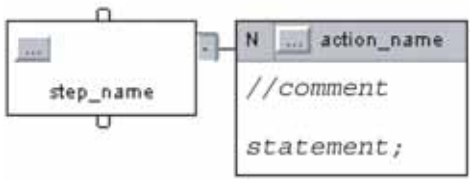
Одним из самых простых способов программирования операции является запись алгоритма в качестве структурированного текста в пределах данной операции. Когда операция включается, контроллер выполняет структурированный текст.



1. Дважды щелкните в текстовой области данной операции.

2. Введите с клавиатуры структурированный текст.

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

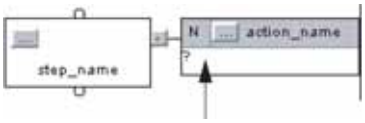


За более подробной информацией обращайтесь:

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

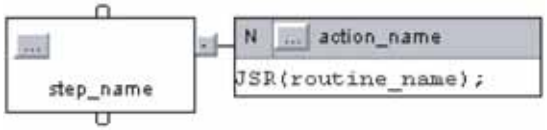
Вызов подпрограммы

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)* .

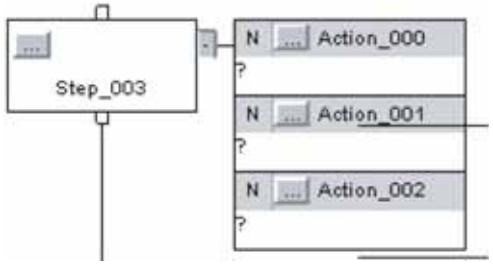
2. Выберите процедуру, которую вы хотите вызвать.
3. Для передачи параметра в процедуру щелкните на пустом окне *Input Parameters (Ввод параметров).* Затем используйте стрелку вниз для выбора тега, содержащего этот параметр.
4. Для получения параметра от процедуры, щелкните на пустом окне *Return Parameters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

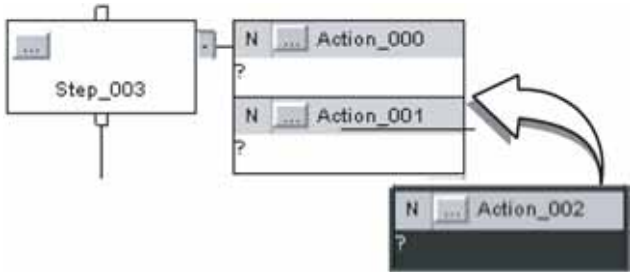
1. Action_000

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

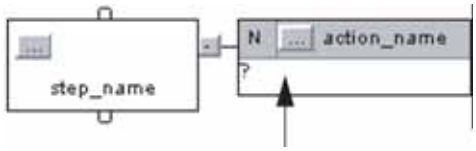
Добавить комментарии на языке структурированного текста

Для форматирования комментариев используйте следующую таблицу:

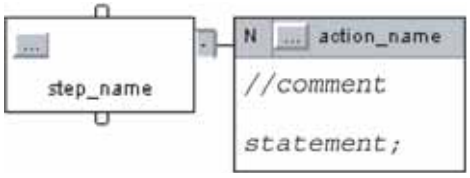
Чтобы добавить комментарий:	Используйте один из следующих форматов:
На одной строке	`//comment`
В конце строки структурированного текста	`(*comment*)` `/*comment*/`
В пределах строки структурированного текста	`(*comment*)` `/*comment*/`
Если занимает больше одной строки	`(*start of comment . . . end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

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

Для ввода комментария:

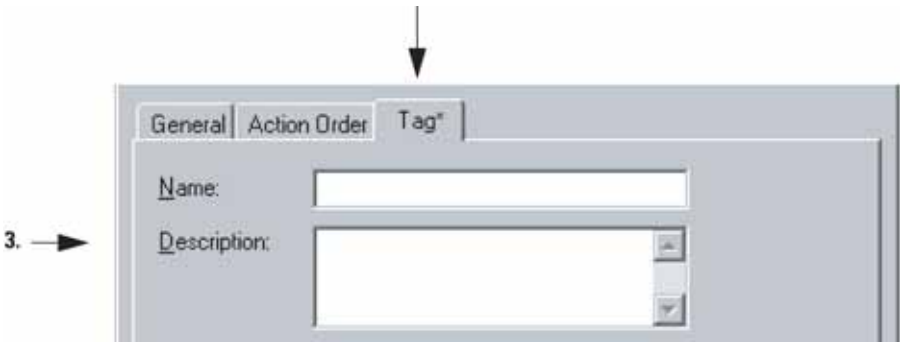


1. Дважды щелкните в текстовой области данной операции.
2. Введите с клавиатуры комментарии.
3. Чтобы закрыть окно текстового ввода, нажмите [Ctrl] + [Enter].



Добавить описания тега

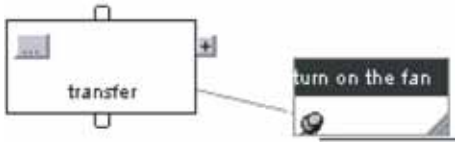
1. Щелкните на кнопке ... данного элемента.
2. Щелкните на закладке *Tag (Тег)*.



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:

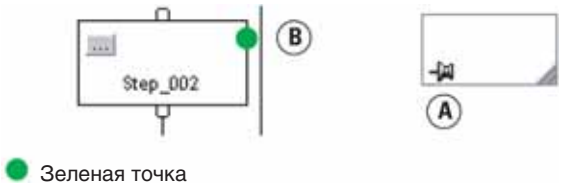


1. Щелкните на 

   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

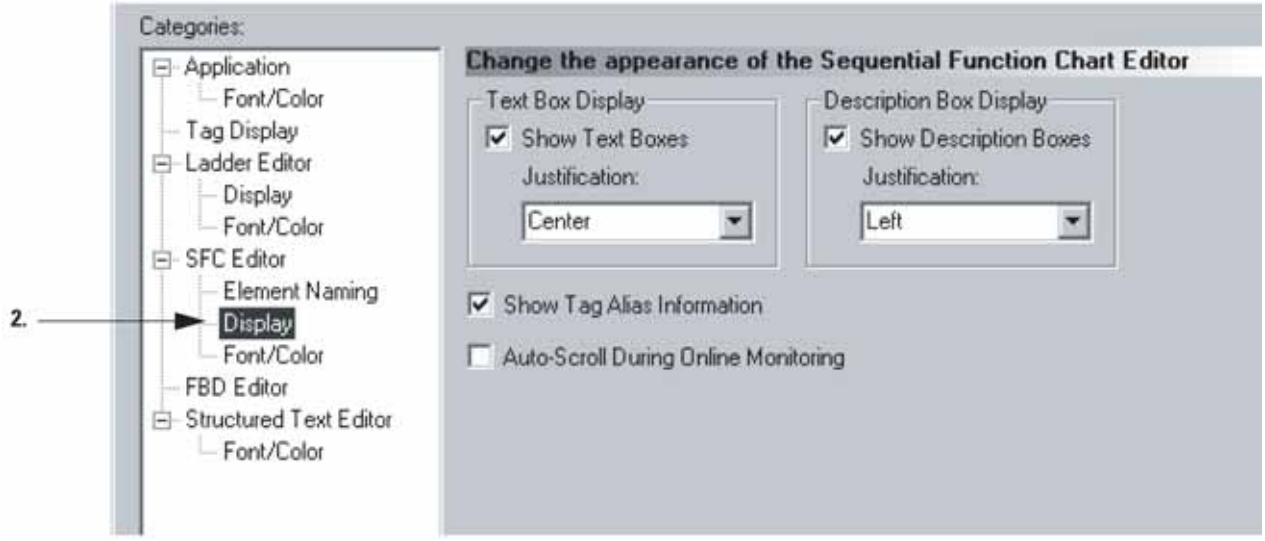
В вашем распоряжении имеются опции, которые позволяют показать или спрятать текстовые окна или описания тегов. Если вы выбрали показ описаний, ПФС показывает описания только для шагов, переходов, останова (а не операций).

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.

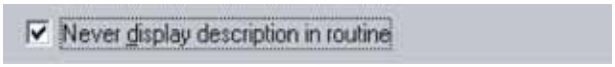
Если вы хотите:	То:
Показать текстовые окна или описания	Установите соответствующий флажок
Спрятать текстовые окна или описания	Снимите соответствующий флажок

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).*



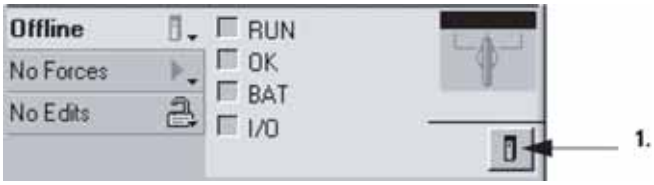
3. Выберите [OK] .

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

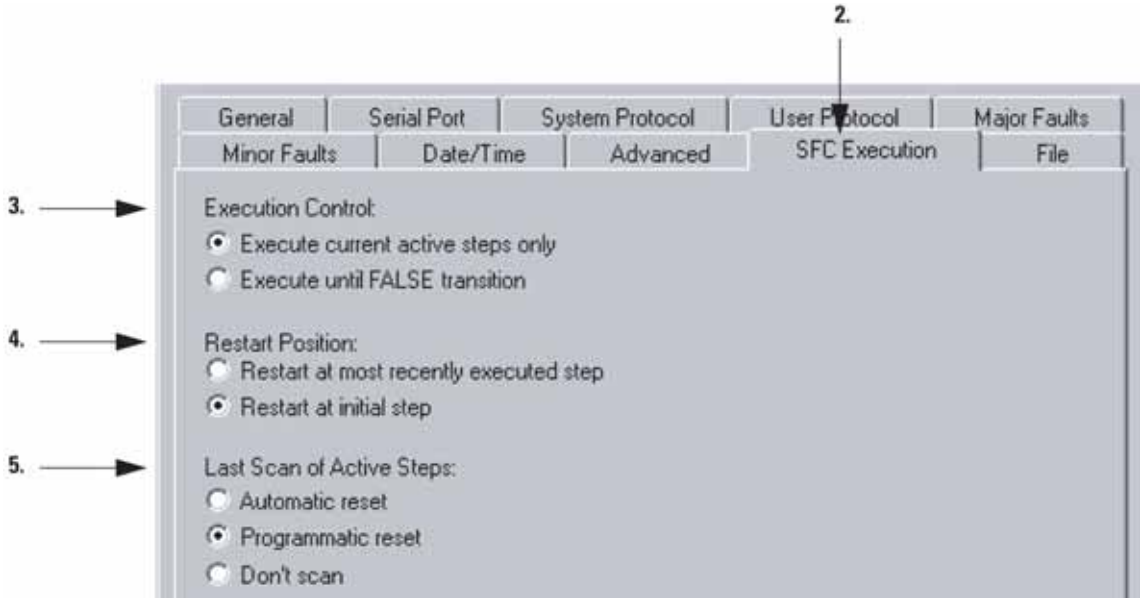
Конфигурирование выполнения ПФС

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:

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 структурированногĈ 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

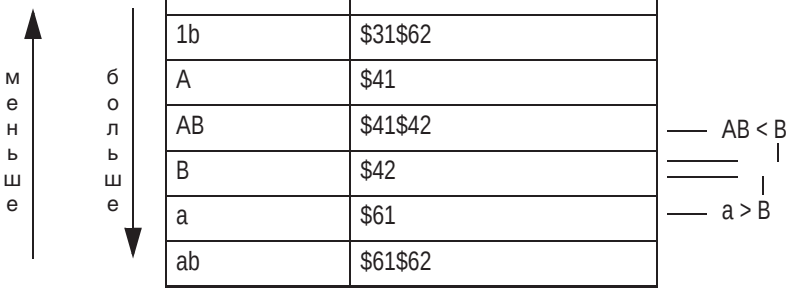
Например:

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.

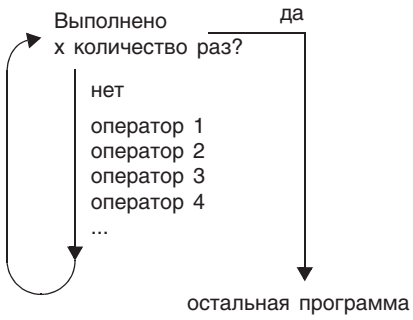
Описание: Синтаксис:

```
FOR count := initial_value
   TO final_value
   BY increment
   DO
   <statement>;
   IF bool_expression THEN
       EXIT;
   END_IF;
END_FOR;
```

необязательно — BY increment: Если вы не задаете приращение, значение увеличивается на 1.

необязательно — IF bool_expression THEN ... END_IF;: 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:

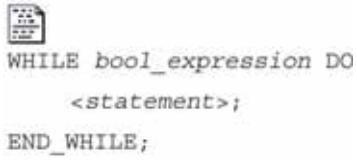
Если вы хотите:	Введите:

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:



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:

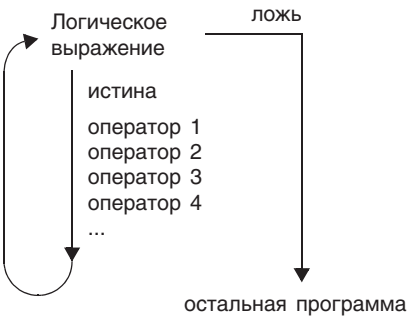
```
WHILE bool_expression1 DO
   <statement>;

   IF bool_expression2 THEN
     EXIT;
   END_IF;

END_WHILE;
```

Операторы для выполнения, пока *bool_expression1* имеет значение "истина".

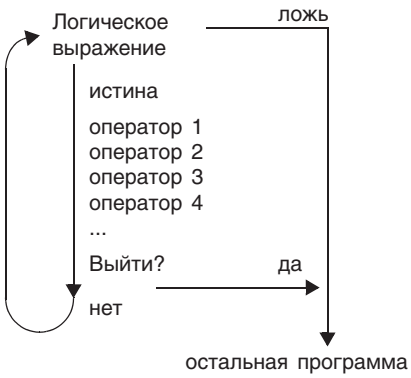
Необязательно

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);`

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

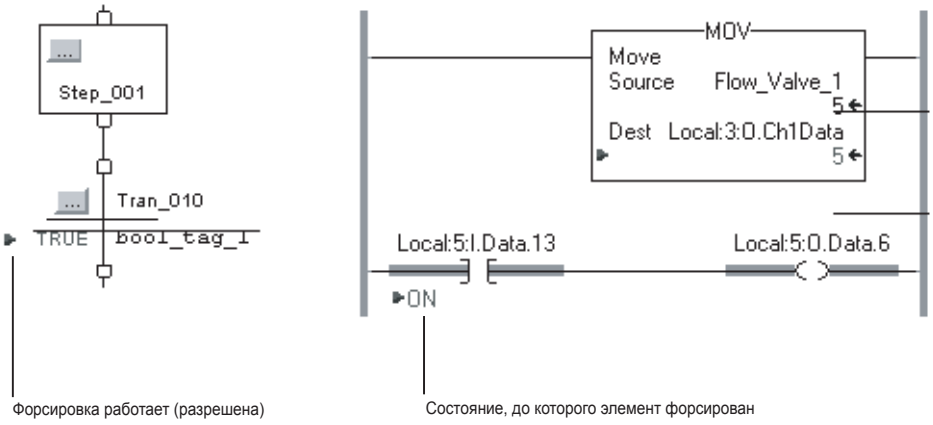
Для того чтобы форсировка заработала, она должна быть разрешена. Вы можете разрешить или запретить форсировку только на уровне контроллера.

- 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

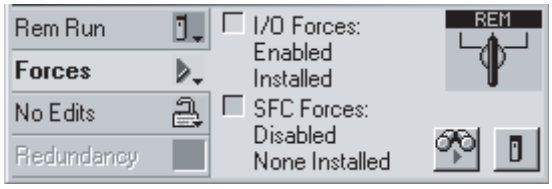
Перед использованием форсировки проверьте состояние форсировок контроллера. Вы можете проверить состояние форсировок следующими способами:

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.

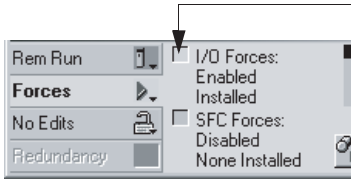
Добавление форсировки ввода\вывода

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

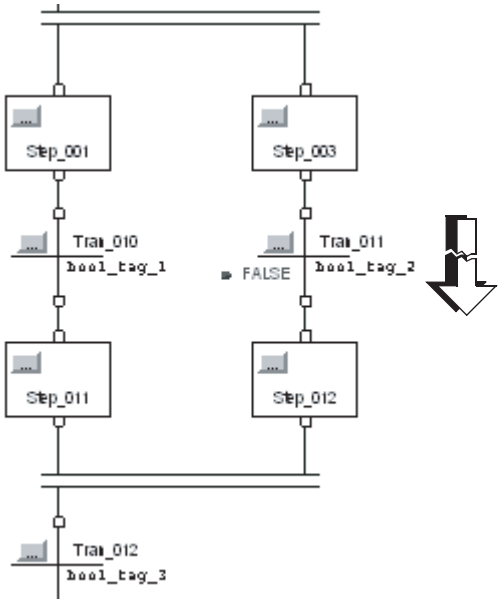
Форсировка перехода

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.

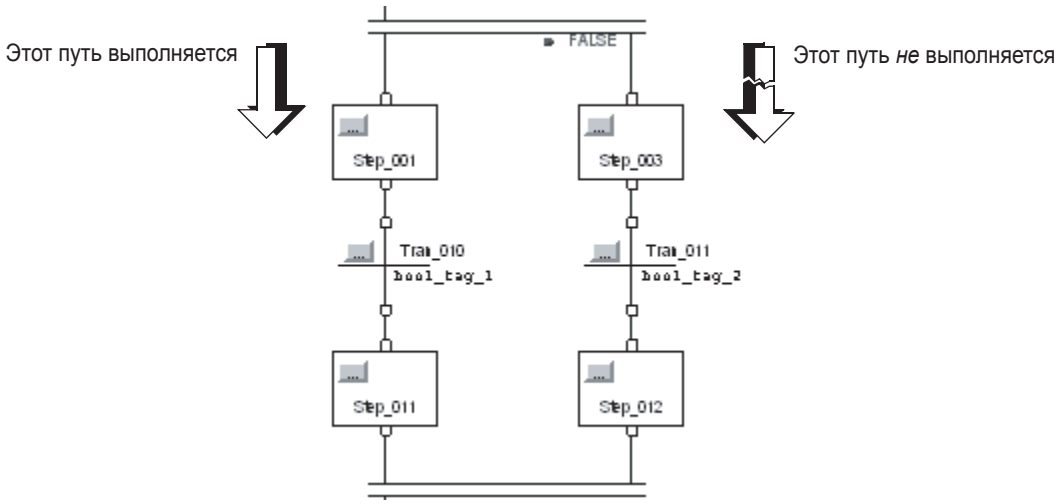


Например, для выхода из этой ветви ПФС должна быть способна:

- 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.

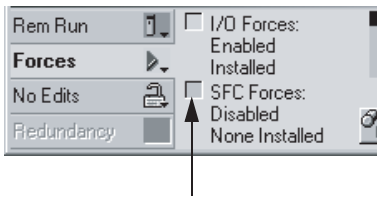
Добавить форсировку ПФС

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.

A

action (действие) 6-19
add (добавление) 6-16
assign order (назначение порядка) 6-22
assign qualifier (назначение определителя) 6-17
boolean (булево) 5-20
choose between boolean and non-boolean (выбор между булевым и небулевым) 5-18
configure (конфигурирование) 6-17
data type (тип данных) 5-20
non-boolean (небулево) 5-18
program (программа) 5-18, 6-19
qualifier (определитель) 5-23
rename (переименование) 6-16
reset (сброс) 5-42
store (сохранение) 5-42
use expression (использование выражения) 6-18
use of structured text (использование структурированного текста) 6-19
alarm (сигнализация)
sequential function chart (последовательная функциональная схема) 5-28, 6-12
arithmetic operators (арифметические операторы)
structured text (структурированный текст) 7-6
ASCII
structured text assignment (присвоение структурированного текста) 7-4
assignment (присвоение)
ASCII character (символ ASCII) 7-4
non-retentive (без сохранения) 7-3
retentive (с сохранением) 7-2
automatic reset (автоматический сброс)
sequential function chart (последовательная функциональная схема) 5-38

B

bitwise operators (побитовые операторы)
structured text (структурированный текст) 7-10
BOOL expression (выражение BOOL)
sequential function chart (последовательная функциональная схема) 5-26, 6-14
structured text (структурированный текст) 7-4
boolean action (булево действие) 5-20, 6-19
program (программа) 5-20
branch (ветвь)
sequential function chart (последовательная функциональная схема) 5-12, 6-5, 6-6

C

CASE 7-16
comments (комментарии)
structured text (структурированный текст) 7-28
configure (конфигурирование)
action (действия) 6-17
alarm (сигнализаци) 6-12
execution of sequential function chart (выполнения последовательной функциональной схемы) 5-50, 6-28
step (шага) 6-11
construct (конструкция)
structured text (структурированного текста) 7-12

D

data (данные)
force (форсировка) 14-6, 14-8
description (описание)
structured text (структурированного текста) 7-28
disable (запрещение)
force (форсировки) 14-3, 14-13
document (документирование)
sequential function chart (последовательной функциональной схемы) 6-23
structured text (структурированного текста) 7-28
documentation (документация)
show or hide in sequential function chart (показывать или скрыть в последовательной функциональной схеме) 6-26
don't scan (не сканировать)
sequential function chart (последовательную функциональную схему) 5-34

E

enable (разрешение)
force (форсировки) 14-2
enter (ввод)
action (действия) 6-16
selection branch (ветви выбора) 6-6
sequential function chart (последовательной функциональной схемы) 6-3
simultaneous branch (одновременной ветви) 6-5
EOT instruction (инструкция EOT) 5-27
execution (выполнение)
sequential function chart (последовательной функциональной схемы) 5-51, 6-28
execution order (порядок выполнения)
function block diagram (функциональной блок-схемы) 9-5
expression (выражение)
BOOL expression (выражение BOOL)
sequential function chart (последовательная функциональная схема) 5-26, 6-14
structured text (структурированный текст) 7-4

numeric expression (численное выражение)
sequential function chart (последовательная функциональная схема) 6-12, 6-18
structured text (структурированный текст) 7-4
order of execution (порядок выполнения)
structured text (структурированный текст) 7-10
structured text (структурированный текст)
arithmetic operators (арифметические операторы) 7-6
bitwise operators (побитовые операторы) 7-10
functions (функции) 7-6
logical operators (логические операторы) 7-9
overview (обзор) 7-4
relational operators (операторы отношения) 7-7

F

FOR...DO 7-19
force (форсировка)
disable (запрещение) 14-3, 14-13
enable (разрешение) 14-2
LED (светодиодный индикатор) 14-4
monitor (контроль) 14-4
options (опции) 14-6
remove (удаление) 14-3, 14-13
safety precautions (меры предосторожности) 14-2
sequential function chart (последовательная функциональная схема) 14-9, 14-12
tag (тег) 14-6, 14-8
function block diagram (функциональная блок-схема)
force a value (форсировать значение) 14-1
functions (функции)
structured text (структурированный текст) 7-6

I

IF...THEN 7-13

J

jump (переход)
sequential function chart (последовательная функциональная схема) 5-17

L

ladder logic (релейная логика)
force a value (форсировка значения) 14-1
override a value (замена значения) 14-1
last scan (последнее сканирование)
sequential function chart (последовательная функциональная схема) 5-32
LED (Светодиодный индикатор)
force (форсировка) 14-4
logical operators (логические операторы)
structured text (структурированный текст) 7-9

M

main routine (главная процедура)
use of sequential function chart (использование последовательной функциональной схемы) 5-6
mark as Boolean (обозначить как булево) 6-19
math operators (математические операторы)
structured text (структурированный текст) 7-6
monitor (контроль)
forces (форсировки) 14-4

N

numeric expression (численное выражение) 6-12, 6-18, 7-4

O

operators (операторы)
order of execution (порядок выполнения) structured text (структурированный текст) 7-10
order of execution (порядок выполнения)
structured text expression (выражения структурированного текста) 7-10

P

pause an SFC (приостановка ПФС) 5-51
periodic task (периодическая задача)
application for (приложение для) 5-5
postscan (пост-сканирование)
sequential function chart (последовательная функциональная схема) 5-32
structured text (структурированный текст) 7-3
priority (приоритет)
selection branch (ветвь выбора) 6-8
program (программа)
action (действие) 5-18, 6-19
boolean action (булево действие) 5-20
transition (переход) 6-14
programmatic reset option (опция программного сброса) 5-35

Q

qualifier (определитель)
assign (присвоение) 6-17
choose (выбор) 5-23

R

relational operators (операторы отношения)
structured text (структурированный текст) 7-7
remove (удаление)
force 14-3, 14-13
rename (переименование)
action (действия) 6-16
step (шага) 6-11
transition (перехода) 6-14
REPEAT...UNTIL 7-25
reset (сброс)
action (действия) 5-42
SFC (ПФС) 5-46
reset an SFC (сброс ПФС) 5-49, 5-51
restart (перезапуск)
sequential function chart (последовательная функциональная схема) 5-46
routine (процедура)
as transition (как переход) 5-27
nest within sequential function chart (вложение в последовательную функциональную схему) 5-49
verify (проверка) 6-29

S

selection branch (ветвь выбора)
assign priorities (назначение приоритетов) 6-8
create (создание) 6-6
overview (обзор) 5-15
sequential function chart (последовательная функциональная схема)
action (действие)
assign order (задание порядка) 6-22
call a subroutine (вызов подпрограммы) 6-21
configure (конфигурирование) 6-17
enter (ввод) 6-16
overview (обзор) 5-18
program (программы) 6-19
rename (переименование) 6-16
use of boolean action (использование булева действия) 5-20
automatic reset option (опция автоматического сброса) 5-38
boolean action (булево действие) 5-20
call a subroutine (вызов подпрограммы) 6-21
configure execution (конфигурирование выполнения) 6-28
define tasks (определение задач) 5-5
document (документирование) 6-23
don”t scan option (опция «не сканировать») 5-34
enter a new element (ввод нового элемента) 6-3
execution (выполнение)
configure (конфигурирование) 5-50
diagrams (схемы) 5-51
pause (приостановка) 5-51
force element (элемент форсировки) 14-1, 14-9, 14-12
last scan (последнее сканирование) 5-32
nest (вложение) 5-49
numeric expression (численное выражение) 6-12, 6-18
organize a project (организация проекта) 5-6
organize steps (организация шагов) 5-12
pause an SFC (приостановка ПФС) 5-51
programmatic reset option (опция программного сброса) 5-35
qualifier (определитель) 5-23
reset (сброс)
data (данных) 5-32
SFC (ПФС) 5-46, 5-49, 5-51
restart (перезапуск) 5-46
return to previous step (возврат к предыдущему шагу) 6-9
selection branch (ветвь выбора)
assign priorities (назначение приоритетов) 6-8
create (создание) 6-6
overview (обзор) 5-15
sequence (последовательность) 5-14
show or hide documentation (показать или спрятать документацию) 6-26
simultaneous branch (одновременная ветвь)
create (создание) 6-5
overview (обзор) 5-16
step (шаг)
configure (конфигурирование) 6-11
define (определение) 5-6
organize (организация) 5-12
overview (обзор) 5-6
rename (переименование) 6-11
step through (шаг через)
simultaneous branch (одновременную ветвь) 14-9
transition (переход) 14-9
step through simultaneous branch (шаг через одновременную ветвь) 14-9
step through transition (шаг через переход) 14-9
stop (стоп) 5-45
text box (текстовое окно) 6-25
transition (переход)
overview (обзор) 5-24
program (программа) 6-14
rename (переименование) 6-14
wire (связь) 5-17
SFC_ACTION structure (структура SFC_ACTION) 5-20
SFC_STEP structure (структура SFC_STEP) 5-8
SFC_STOP structure (структура SFC_STOP) 5-47
SFP instruction (инструкция SFP) 5-51

SFR instruction (инструкция SFR) 5-46, 5-49, 5-51
simultaneous branch (одновременная ветвь) 5-16
enter (ввод) 6-5
force (форсировка) 14-9, 14-12
step through (шаг через) 14-9
slot number (номер слота) 1-3
source key (ключ источника) 18-1
status (состояние)
force (форсировки) 14-4
step (шаг)
add action (добавление действия) 6-16
alarm (сигнализация) 5-28
assign preset time (задание уставки по времени) 6-11
configure (конфигурирование) 6-11
configure alarm (конфигурирование сигнализации) 6-12
data type (тип данных) 5-8
define (определение) 5-6
organize in sequential function chart (организация в последовательной функциональной схеме) 5-12
rename (переименование) 6-11
selection branch (ветвь выбора) 5-15
sequence (последовательность) 5-14
simultaneous branch (одновременная ветвь) 5-16
timer (таймер) 5-28
step through (шаг через)
simultaneous branch (одновременную ветвь) 14-9
transition (переход) 14-9
stop (стоп)
data type (тип данных) 5-47
sequential function chart (последовательная функциональная схема) 5-45
store (сохранение)
action (действия) 5-42
string (строка)
evaluation in structured text (оценка в структурированном тексте) 7-8
string data type (строковый тип данных)
create (создание) 12-8
structure (структура)
SFC_ACTION 5-20
SFC_STEP 5-8
SFC_STOP 5-47
structured text (структурированный текст)
arithmetic operators (арифметические операторы) 7-6
assign ASCII character (присвоение символа ASCII) 7-4
assignment (присвоение) 7-2
bitwise operators (побитовые операторы) 7-10
CASE 7-16
comments (комментарии) 6-23, 7-28
components (компоненты) 7-1
contructs (конструкции) 7-12
evaluation of strings (оценка строк) 7-8
expression (выражение) 7-4
FOR...DO 7-19
force a value (форсировка значения) 14-1
functions (функции) 7-6
IF...THEN 7-13
in action (в действии) 6-19
logical operators (логические операторы) 7-9
non-retentive assignment (присвоение без сохранения) 7-3
numeric expression (численное выражение) 7-4
relational operators (операторы отношения) 7-7
REPEAT...UNTIL 7-25
WHILE...DO 7-22

T

tag (тег)
force (форсировка) 14-6, 14-8
task (задача)
define (определение) 5-5
text box (текстовое окно)
sequential function chart (последовательная функциональная схема) 6-25
show or hide in sequential function chart (показать или спрятать в последовательной функциональной схеме) 6-26
transition (переход)
BOOL expression (выражение BOOL) 5-26
call subroution (вызов подпрограммы) 6-15
choose program method (метод выбора программы) 5-26
define (определение) 5-24
EOT instruction (инструкция EOT) 5-27
force (форсировка) 14-9, 14-12
program (программа) 6-14
rename (переименование) 6-14
step through (шаг через) 14-9
use of a subroutine (использование подпрограммы) 5-27

V

verify (проверка)
routine (процедуры) 6-29

W

WHILE...DO 7-22
wire (связь)
sequential function chart (последовательная функциональная схема) 5-17, 6-9

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 SFC and ST Programming Languages

Cat. No. Excerpt from the *Logix5000 Controllers Common Procedures*, publication 1756-PM001 Pub. No. 1756-PM003G-EN-E (excerpt of 1756-PM001G) Pub. Date March 2004 Part No. 957867-43

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-PM003G-EN-E (excerpt of 1756-PM001G) - March 2004 PN 957867-43
Supersedes Publication 1756-PM003F-EN-E - 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-PM003G-EN-E (excerpt of 1756-PM001G) - March 2004