

TM835

# Использование языков программирования ST и SFC в APROL



## **Требования**

Учебные модули	TM213 – Automation Runtime TM242 – Последовательные функциональные схемы (SFC) TM246 – Язык структурированного текста (ST) SEM841.5 – Семинар APROL Process Control: Основы 1
Программная часть	APROL Automation Runtime SuSE LINUX
Аппаратная часть	1 управляющий компьютер, 1 контроллер

**Оглавление**

1 Введение.....	4
1.1 Цель обучения.....	5
2 Общая информация о ST.....	6
2.1 Особенности структурированного текста.....	6
2.2 Основные элементы и группы команд.....	7
3 Общая информация о SFC.....	9
3.1 Основные функции SFC.....	10
3.2 Различие между простыми шагами и шагами IEC.....	14
3.3 Какими особыми функциями обладает SFC в APROL?.....	15
3.4 Чем SFC отличается от остальных языков IEC?.....	16
4 ST в разработке проектов.....	17
4.1 Введение и функции редактора.....	17
4.2 Создание программ ST.....	18
4.3 Работа с редактором ST.....	19
4.4 Отладка программ ST.....	27
5 SFC в разработке проектов.....	30
5.1 Введение и функции редактора.....	30
5.2 Создание программы SFC.....	31
5.3 Работа с редактором SFC.....	32
5.4 Диагностические функции в SFC.....	49
6 ST в разработке библиотек.....	65
6.1 Введение и функции редактора.....	65
6.2 Создание функционального блока ST.....	65
6.3 Основные данные функционального блока.....	67
6.4 Использование контактов и переменных.....	68
6.5 Использование блоков Automation Studio (AS) в функциональном блоке ST.....	69
6.6 Практическое задание – кран.....	72
7 SFC в разработке библиотек.....	73
7.1 Введение и функции редактора.....	73
7.2 Создание функционального блока SFC.....	73
7.3 Основные данные функционального блока SFC.....	74
7.4 Графическая настройка функционального блока.....	75
7.5 Использование контактов и переменных.....	76
7.6 Вызов функций и функциональных блоков.....	78
7.7 Практическое задание – управление емкостью.....	78
8 Заключение.....	80
9 Приложение.....	81
9.1 Пример программ ST для решения задач из текста модуля.....	81
9.2 Пример программ SFC для решения задач из текста модуля.....	84

# Введение

## 1 Введение

Вы уже ознакомились с основами системы управления технологическими процессами APROL от B&R и обладаете информацией как о ее структуре, так и о том, как осуществляется ввод оборудования в эксплуатацию.

Прежде чем создать логику работы оборудования, вам придется иметь дело с языками программирования, доступными в системе управления технологическими процессами. Язык структурированного текста (Structured Text, ST) – это быстрый и эффективный способ программирования в области автоматизации. Язык структурированного текста – это язык программирования высокого уровня, который вобрал в себя основные элементы языков Basic, Pascal и ANSI-C.

Возможно, вы уже знаете о простых стандартных структурах, командах, ключевых словах и синтаксисе ST из учебного модуля Automation Studio TM246 и семинара SEM246.1.

Во второй части будет подробно описан язык последовательных функциональных схем (SFC). Этот язык программирования состоит из графических элементов и является шагово-ориентированным, а потому прекрасно подходит для описания последовательных этапов процесса на предприятии.

Помимо своей простоты SFC обладает еще одним преимуществом – он содержит множество возможностей для графического анализа, с которыми мы ознакомимся позже. Для изучения элементов и функций языка последовательных функциональных схем мы рекомендуем изучить учебный модуль по Automation Studio TM242.



Вы уже изучили учебные модули Automation Studio по ST и SFC или знакомы с этим видом программирования?

Из данного модуля вы узнаете, как происходит создание в APROL частей проекта, использующих ST и SFC. Помимо создания программ и функциональных блоков здесь будут объяснены различные способы отладки и поиска ошибок. Полученные знания будут закреплены при помощи примеров и упражнений.

## 1.1 Цель обучения

Из данного модуля вы узнаете, как работать со структурированным текстом и последовательными функциональными схемами. В APROL языки ST и SFC можно использовать, как в контексте программ проекта, так и создавать функциональные блоки на языках ST и SFC в контексте библиотек.

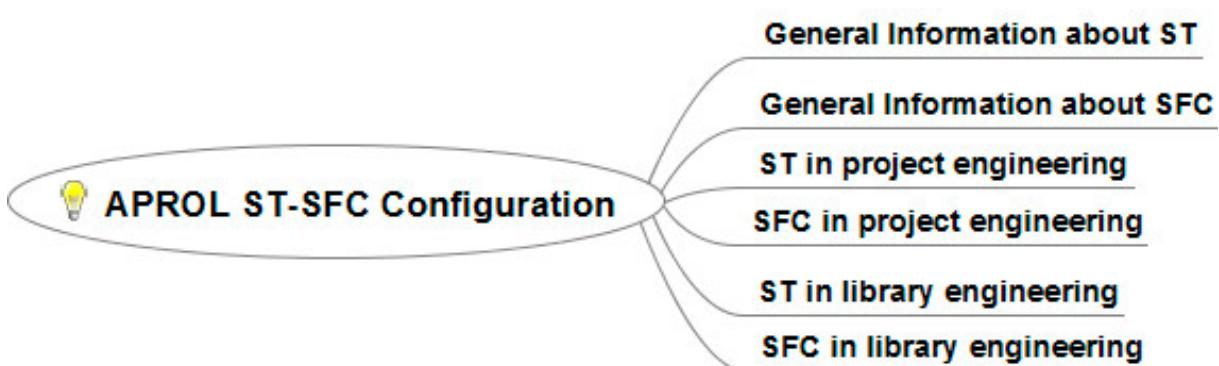
В начале модуля мы повторим основные и наиболее важные функции ST и SFC.

Исходя из этого, модуль делится на две части. В первой рассматривается язык структурированного текста. Здесь будет объяснено, как создавать части проекта на основе ST, как объявлять и использовать переменные, а также будут рассмотрены другие функции редактора ST.

Язык последовательных функциональных схем рассматривается во второй части. Вы узнаете, как создать части проекта на основе SFC, а также как объявлять и использовать переменные в них. Редактор SFC также содержит множество других функций, которые будут рассмотрены при создании последовательности технологического процесса.

Также вы узнаете о методах диагностики программ ST и SFC. Для этих целей в системе существует режим мониторинга, программа SFCViewer и различные отчеты SFC.

В данном документе содержится множество снимков экрана, которые помогут вам в разработке и настройке собственной системы. Выполнение упражнений так же поможет оценить функционал системы.



# Общая информация о ST

## 2 Общая информация о ST

### 2.1 Особенности структурированного текста

#### Общая информация

ST – это язык высокого уровня на основе текста, предназначенный для программирования систем автоматизации. Простые стандарты построения обеспечивают быстрое и эффективное программирование. ST использует множество характерных для языков высокого уровня возможностей, включая переменные, операторы, функции и элементы, для управления исполнением программы.

ST – язык программирования, стандартизированный по IEC<sup>1</sup>.

#### Свойства

**Язык структурированного текста характеризуется следующими свойствами:**

- Язык программирования высокого уровня
- Структурированное программирование
- Простые в использовании стандартные конструкции
- Быстрое и эффективное программирование
- Не требует объяснения и гибок в использовании
- Похож на PASCAL
- Соответствует стандарту IEC 61131-3

#### Возможности

**В APROL поддерживаются следующие функции:**

- Использование коннекторов, дискретных и аналоговых вводов и выводов
- Логические операции
- Логические выражения сравнения
- Арифметические операции
- Операторы условий
- Оператор выбора
- Циклы
- Создание функциональных блоков
- Вызов функций и функциональных блоков из библиотек
- Отслеживание значений переменных в реальном времени (отладка)

<sup>1</sup> IEC 61131-3 является принятым во всем мире стандартом для программируемых логических контроллеров. Помимо языка структурированного текста в нем также описаны языки последовательных функциональных схем (SFC), лестничных схем (Ladder Diagram), списков инструкций (Instruction List) и непрерывных функциональных схем (Continuous Function Chart).

## 2.2 Основные элементы и группы команд

Чтобы не повторять информацию, содержащуюся в другом модуле, мы предлагаем вам получить подробные сведения об основных командах языка структурированного текста, ознакомившись с учебным модулем Automation Studio TM246.

В этом модуле основные данные и важные вопросы будут рассмотрены и повторены лишь частично.

Назначение в ST состоит из переменной слева, которой назначается результат вычислений из правой части с помощью оператора назначения ":". Все назначения должны быть завершены точкой с запятой ";".



```
Result := ProcessValue * 2; (*Result ← (ProcessValue * 2) *)
```

Таблица 1: Назначение производится слева направо

После обработки строки кода значение переменной "Result" станет в два раза больше значения "ProcessValue".

### В ST различают следующие группы команд:

- **Булевые операции**



Булевые операции могут быть совмещены любым образом. Результатом выражения могут являться только значения TRUE (логическая 1) или FALSE (логический 0).

```
Variable_1 := Variable_2 AND NOT Variable_3;
Variable_1 := Variable_2 OR Variable_3;
Variable_1 := Variable_2 XOR Variable_3;
```

- **Арифметические операции**



Ощутимым преимуществом языков программирования высокого уровня является простота обработки арифметических операций, таких как сложение, вычитание, умножение и т. д.

```
Variable_1 := Variable_2 + Variable_3;
Variable_1 := Variable_2 - Variable_3;
Variable_1 := (Variable_2 * Variable_3) / (Variable_4 + 1);
```

- **Операторы сравнения и решения**



Для сравнения нескольких переменных используются простые конструкции. Операторы сравнения возвращают значение TRUE или FALSE и в основном используются в качестве условий в решениях, таких как IF, ELSIF, WHILE и UNTIL.

```
IF a > b THEN
    Result := 1;
ELSIF (a > b) AND (a < c) THEN
    Result := 2;
ELSE
    Result := 3;
END_IF
```

# Общая информация о ST

- Конечные автоматы – оператор Case



Оператор CASE следует использовать вместо IF в случае, если при помощи IF выполняется проверка одной и той же переменной. Оператор CASE выполняет пошаговое сравнение переменной с несколькими значениями. Если в процессе одного из сравнений обнаруживается соответствие, то выполняется определенное действие. Если ни одно из значений не подходит, выполняется программный код, описанный после ELSE.

```
CASE StepVariable OF
 1,5:           Result := 100;
 2:             Result := 500;
 3,4,6..10:     Result := 1000;
 ELSE
   Result := 0;
END_CASE
```

- Циклы



Использование циклов имеет смысл тогда, когда часть кода необходимо повторить в пределах одного цикла. Блок кода цикла будет повторяться до тех пор, пока не будет достигнуто условие завершения.

Различают циклы с предусловием и циклы с постусловием: цикл с предусловием (FOR, WHILE) проверяет условие завершения до начала выполнения цикла, а цикл с постусловием (REPEAT) проверяет условие завершения в конце цикла, таким образом выполняя код как минимум один раз.

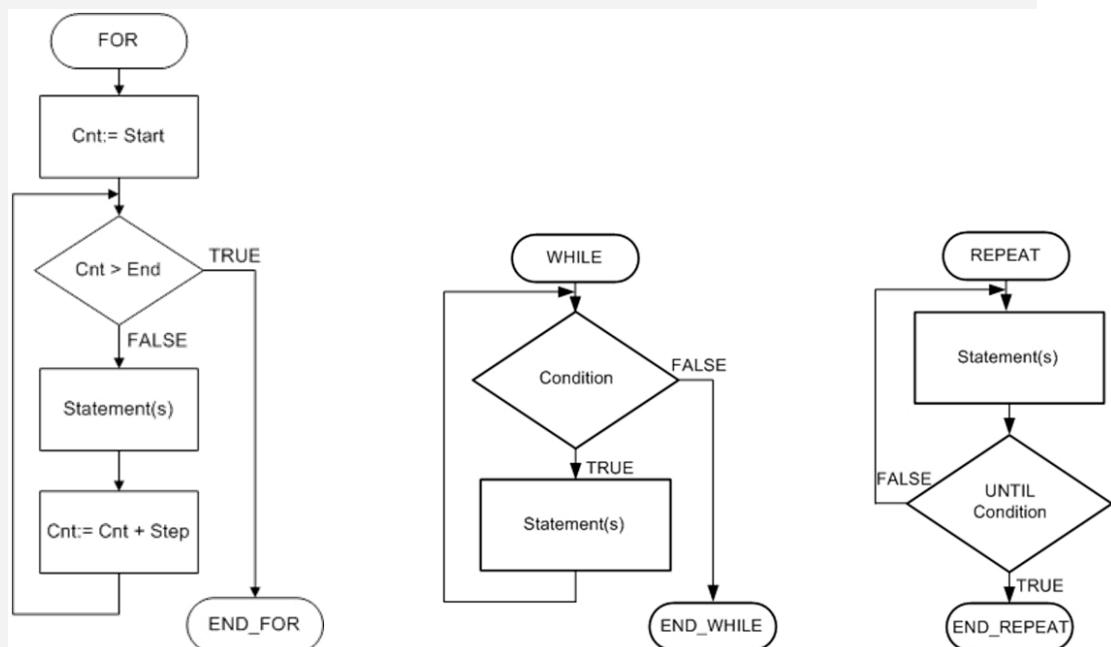


Рисунок 1: Различные виды циклов

```
//Beispiel für eine FOR-Schleife:
FOR i:= StartVal TO StopVal BY StepVal DO
  Result := Result + 1;
END_FOR
```

## 3 Общая информация о SFC

Язык последовательных функциональных схем – это язык визуального программирования, который дает возможность организовать последовательное выполнение различных разделов и шагов программы. Этот язык используется тогда, когда необходимо определить шаги программы и переходы к отдельным шагам. Существует возможность даже выполнять параллельные последовательности в пределах одной программы.

SFC – язык программирования, стандартизированный по IEC<sup>2</sup>. В шаге могут вызываться действия, описание которых выполняется при помощи языка структурированного текста.



В дополнение к действиям, ST также можно использовать при программировании шагов и условий перехода SFC.

Язык SFC как нельзя лучше подходит для использования с конечными автоматами. Возможность одновременного выполнения (параллельные ветви) позволяет легко реализовать сложные концепции. Визуальная структура программы также упрощает процессы документирования и отладки.

### Когда используется SFC?

Язык программирования SFC можно использовать в буквальном смысле везде, где технологический процесс может быть представлен в виде описания последовательности шагов. Конечные аппараты могут быть «переписаны» и внедрены в SFC; обратное действие обычно не представляется возможным.

### Создание крупных приложений

Чтобы получить все преимущества от предназначения SFC – управления последовательностями, – необходимо тщательно обдумать концепцию программного обеспечения, прежде чем начинать создание больших приложений.

Необходимые последовательности рекомендуется реализовывать в программе SFC. Фоновые функции устройств, такие как оценка вводов/выводов, масштабирование значений или управление действиями, подлежат реализации в дополнительной части проекта.

Вызов фоновых функций устройств в SFC осуществляется с помощью определенных интерфейсов; те же интерфейсы содержат состояние функций.

Преимущество этого метода заключается в том, что при изменении фоновых функций устройств (новый датчик, другой привод) не изменяется последовательность. Последовательность действий и фоновые функции можно легко отлаживать и масштабировать, именно благодаря их разделению.



Оставшаяся часть документа посвящена языку последовательных функциональных схем (SFC).

<sup>2</sup> IEC 61131-3 является принятым во всем мире стандартом для программируемых логических контроллеров. Помимо языка последовательных функциональных схем в нем также описаны языки структурированного текста (ST), лестничных схем (Ladder Diagram), списков инструкций (Instruction List) и непрерывных функциональных схем (Continuous Function Chart).

# Общая информация о SFC

## 3.1 Основные функции SFC

Основными компонентами SFC являются шаги (step) и переходы (transition). Шаги отображаются в виде прямоугольников с границами различного вида. Переходы между шагами отображаются в виде полос, условие перехода - пересекает полосу маленьким прямоугольником.

Все шаги и переходы соединяются друг с другом обязательно чередуясь. Другими словами, за шагом всегда следует переход. Шаг и его переход формируют единицу. Последовательное выполнение двух шагов или двух переходов невозможно.

В приведенном здесь примере показана программа, состоящая из трех шагов, связанных с ними переходов и прыжка.

Шаг "FIRST" (Первый) запускается при первичном вызове программы SFC. Он заключен в двойной прямоугольник и всегда вызывается при инициализации программы. Связанный с ним переход имеет значение TRUE, что означает, что шаг инициализации будет вызван только на один цикл.

Программа остается на шаге "HEAT" (Нагрев) до тех пор, пока не станет истинным условие "tooHot" (Слишком горячо). После этого шаг "HEAT" становится неактивным, и программа переходит к шагу "COOL" (Охлаждение). Как только переход "tooCool" (Слишком холодно) принимает значение TRUE, выполняется прыжок, а шаг "HEAT" опять становится активным.

За один цикл программы выполняется только один шаг.

Имена шагов являются символьными, их не надо дополнитель- но объявлять\описывать. Переходы являются переменными типа BOOL или логическими выражениями, которые могут быть написаны на различных языках программирования.

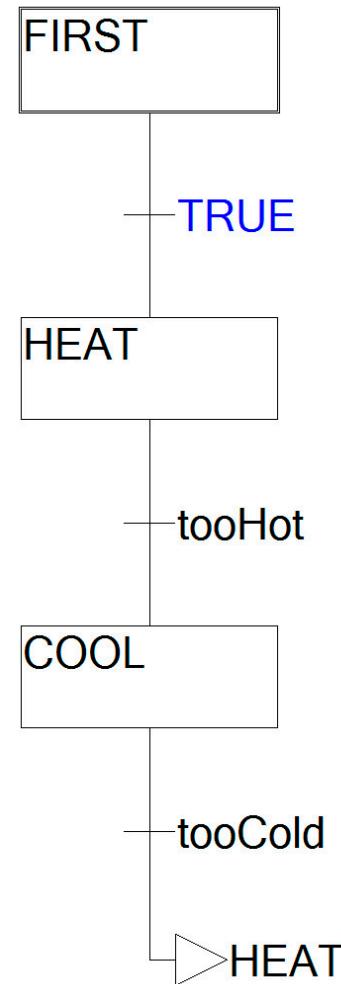


Рисунок 2: Последовательность из двух шагов, одного прыжка и шага инициализации



Шаг и переход формируют единицу. После шага следует переход. Последовательное появление двух или более переходов или шагов невозможно.

Если переход соединяет один шаг с другим, то второй шаг выполняется в следующем цикле программы.

## Шаги SFC

Шаг может быть активным или неактивным. Это отражает ситуацию, в которой команды или действия обрабатываются только в случае, если они активны. Когда шаг становится неактивным, следующий за ним переход принимает решение на основе оценки условия перехода.

Шаги представлены в виде прямоугольников. Если в шаге содержится циклический программный код, в правом верхнем углу прямоугольника отображается черный треугольник. Каждый шаг может (но не должен) содержать входное и выходное действия. Входное действие (E) вызывается тогда, когда шаг становится активным. Выходное действие (X) вызывается после переключения шага в неактивное состояние.

Для шагов также можно настроить мониторинг времени.

Содержимое шага составляется из действий, программирование которых осуществляется на языке ST.

## Переходы SFC

Каждый шаг активируется и деактивируется посредством переходов. Если переход является активным, (TRUE, логическая "1"), соответствующий шаг будет деактивирован, а в следующем цикле будет активирован следующий шаг. Если переход имеет значение FALSE, или логический "0", он становится неактивным, и далее в течение программы не будет выполняться переключение.

Для каждого перехода существует условие перехода, назначенное ему, т. е. правило вычисления, возвращающее булево значение. В APROL программирование условия перехода может быть выполнено только на языке IEC 'ST'.

## Действия SFC

Действия назначаются шагу. Когда шаг становится активным, активными также становятся действия; они выполняются в соответствии с заданными им атрибутами (определителями).

Программирование содержимого действия также выполняется на языке IEC 'ST'.

### Обзор важных определителей действий:

- **N (non-stored)**  
Циклический вызов действия до тех пор, пока шаг остается активным (N)
- **R (overriding reset)**  
Сброс действия, как только до него дойдет очередь выполнения (R).
- **S set (Stored)**  
Действие выполняется и остается активным и в других шагах, пока его не сбросят (S).
- **L (time Limited)**  
Действие ограничено заданным временем (L)
- **D (time Delayed)**  
После квалификатора необходимо указывать продолжительность времени или константой типа TIME, или переменной типа TIME, например, L T # 1h2m3s4ms, L MaxTime. Действие активируется по истечении заданного времени, если блок действия все еще активен.

## Общая информация о SFC

- **P pulses**

Действие выполняется дважды; первый раз, когда шаг становится активным, второй раз, когда не активным. Действие выполняется по истечении заданного времени и остается активным и в других шагах, пока не встретит сброс.

- **SD Stored and time Delayed**

После квалификатора необходимо указывать продолжительность времени или константой типа TIME, или переменной типа TIME, например, D T # 1h2m3s4ms, D MaxTime. Действие активируется по истечении заданного времени, если блок действия все еще активен, и будет активно до сброса.

- **DS Delayed and Stored**

После квалификатора необходимо указывать продолжительность времени или константой типа TIME, или переменной типа TIME, например, SD T # 1h2m3s4ms, SD MaxTime. Действие активируется по истечении заданного времени, если блок действия все еще активен, и будет активно до сброса.

- **SL Stored and time Limited**

После квалификатора необходимо указывать продолжительность времени или константой типа TIME, или переменной типа TIME, например, DS T # 1h2m3s4ms, DS MaxTime. Действие активируется по истечении заданного времени и будет активно до сброса. Это циклический вызов, пока шаг активен (N).



Отдельные шаги в SFC необязательно должны выполняться строго друг за другом. Для повышения гибкости существует возможность добавления параллельных и альтернативных ветвей.

### Параллельная ветвь (одновременная последовательность)

За переходом могут следовать несколько шагов, размещенных параллельно и соединенных двойной линией. После последнего перехода параллельные шаги опять объединяются при помощи двойной горизонтальной линии.

Все одновременные пути в параллельных ветвях (одновременные последовательности) обрабатываются единовременно (т. е. параллельно) сразу после переключения предшествующего им перехода. Оценка перехода, следующего за одновременной последовательностью, производится только в том случае, если последние шаги всех параллельных путей одновременно являются активными.

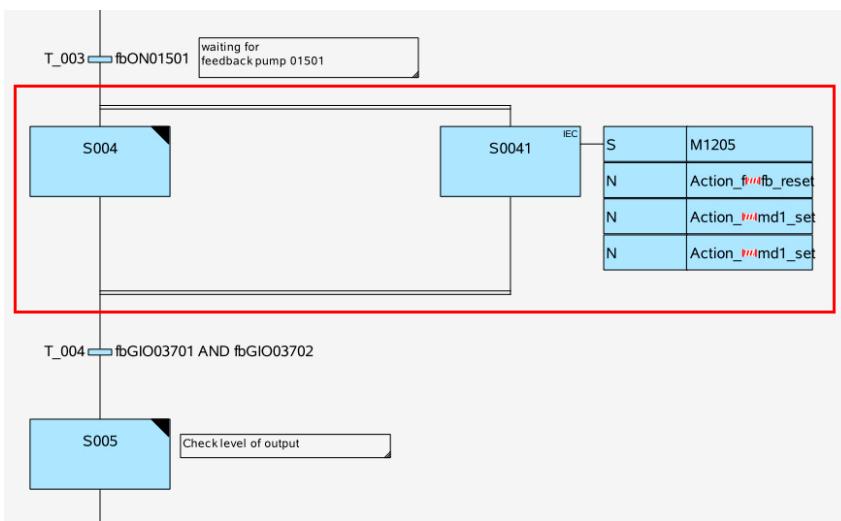


Рисунок 3: Начало одновременных последовательностей

## Альтернативная ветвь

Если последовательность необходимо разделить на несколько различных путей, нужно создать в SFC одну или несколько альтернативных ветвей. В альтернативной ветви активация каждого шага осуществляется при помощи назначенного ему перехода. Таким образом, путь разделяется в зависимости от остатка/альтернатив. Если активными одновременно являются несколько переходов, их оценка осуществляется слева направо, т. е. чем ближе переход к левому краю, тем выше его приоритет.

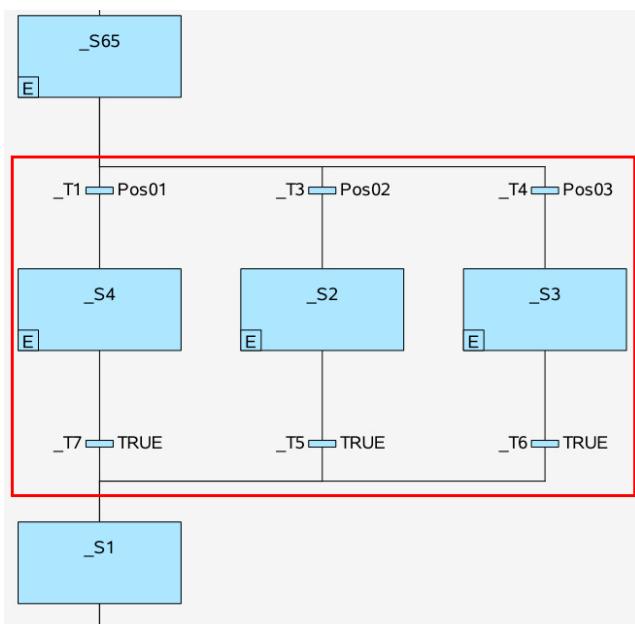


Рисунок 4: Альтернативная ветвь

Альтернативная ветвь может вернуться к общему пути или завершиться прыжком.

# Общая информация о SFC

## Прыжки

Использование прыжков обеспечивает гибкость конструкции последовательностей шагов. Активация прыжка также осуществляется при помощи перехода. Можно выполнить прыжок к любому шагу в сети SFC. Прыжок может быть выполнен только к шагу, но не к переходу.

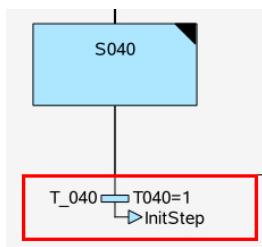


Рисунок 5: Безусловный переход

## 3.2 Различие между простыми шагами и шагами IEC

Мы уже знаем, что шаг может содержать выполняемый программный код. Выполнение кода начинается сразу после активации шага. Существует два типа шагов – «простые» шаги и шаги «IEC». Различия между ними приведены в таблице ниже.

Простой шаг	Шаги IEC
Требуется меньше физической памяти на контроллере	Соответствует стандарту IEC
Возможно 3 действия на шаг (действия Entry (Вход), Cyclic (Цикл) и Exit (Выход))	До 32 действий на шаг (плюс действия Entry и Exit)
Действия ведут себя так же, как действия, назначенные шагам IEC, с определителем "N"	Действиям могут назначаться различные определители
Действия могут использоваться только в одном шаге	Существующие действия могут быть назначены несколько раз различным шагам.

Таблица 2: Различные виды шагов SFC

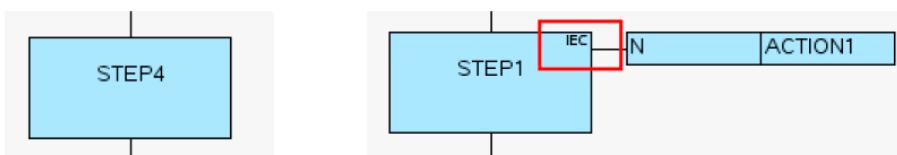


Рисунок 6: Сравнение простых шагов (слева) и шагов IEC (справа)



Редактор SFC обозначает принадлежность шага к одному из указанных типов.

Простые шаги, уже размещенные на функциональной схеме, могут быть преобразованы в шаги IEC, и наоборот.

### 3.3 Какими особыми функциями обладает SFC в APROL?

Части проекта 'ST' и 'SFC' поддерживаются концепцией разработки библиотек APROL для структурированного программирования.

Возможности использования блоков, содержащихся в библиотеке (библиотека CAE) в частях проекта 'CFC', 'ST' и 'SFC' (проект CAE), схематически объяснены ниже.

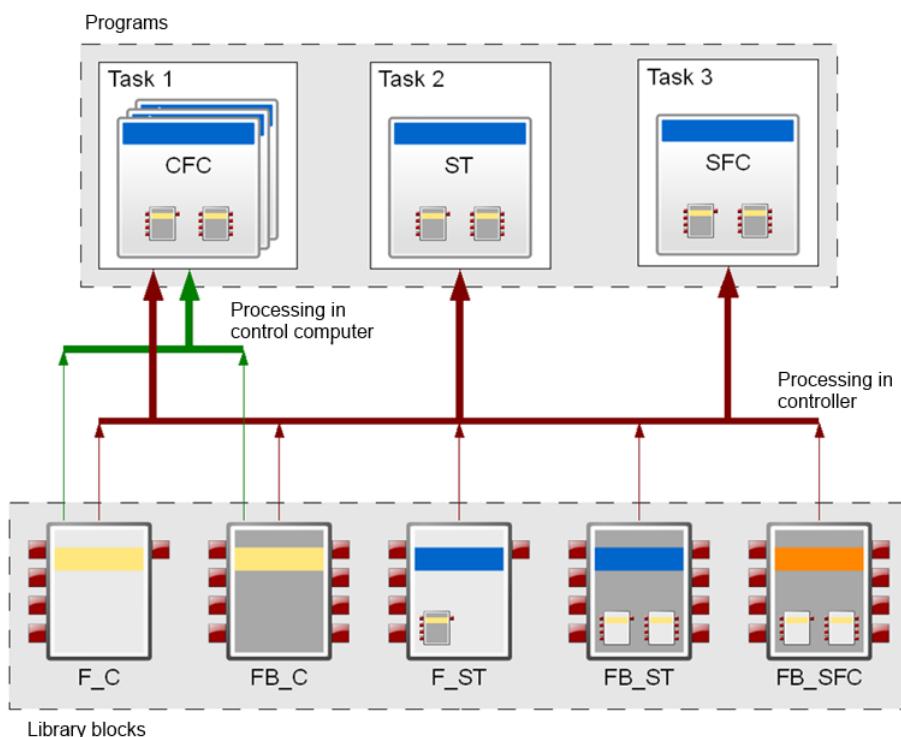


Рисунок 7: Схема возможностей использования блоков

**Легенда: 'F' – функция, 'FB' – функциональный блок, 'C' – язык программирования 'C'**

- Программы SFC и ST по умолчанию могут использоваться только на контроллере.
- Программа SFC или ST может использоваться контроллером только в том случае, если задача контроллера назначена только она одна. Задача, которой назначена программа, **не может являться назначаемой по умолчанию задачей контроллера**.
- Однако несколько программ CFC могут быть назначены одной задаче контроллера.
- В программе ST, SFC или CFC может использоваться любое количество функциональных блоков (ST, SFC или C).

#### Выбор: «программа SFC» или «функциональный блок SFC»

Программа SFC используется в случае, когда последовательное управление, касающееся нескольких частей производственной системы, требуется выполнить только **однократно**.

Функциональный блок SFC используется в случае, когда последовательность действий необходимо повторять многократно. Изменения в функциональный блок вносятся централизованно в библиотеке CAE и автоматически передаются во все экземпляры.

## Общая информация о SFC

### 3.4 Чем SFC отличается от остальных языков IEC?

Языки программирования IEC languages (EN 61131-3:2003)

Язык программирования	Описание
Function block language (язык функциональных блоков)	В APROL называется языком непрерывных функциональных схем (CFC). Содержит логические схемы
Структурированный Текст	В APROL называется языком структурированного текста (ST). Его можно сравнить с языками программирования высокого уровня.
Instruction List (язык списка инструкций)	Язык списка инструкций (IL) можно сравнить с ассемблером.
Ladder Diagram (язык лестничных схем)	Язык лестничных схем (LD) можно сравнить с принципиальной схемой.
Sequential Function Chart (SFC, язык последовательных функциональных схем)	В сравнении с другими языками IEC язык последовательных функциональных схем (SFC) в APROL <b>особенно</b> подходит для моделирования и программирования <b>сложных последовательных процессов</b> . Его можно сравнить с схемой состояний.

#### Дополнительные преимущества языка программирования SFC:

- простое структурирование процессов, которые основываются на последовательности действий;
- подходит для операций, ориентированных по времени и по событию;
- графическая настройка сети SFC;
- модульная организация программы/возможность повторного использования функциональных блоков;
- прост в изучении;
- прост в эксплуатации;
- программы легко масштабируются.

## 4 ST в разработке проектов

### 4.1 Введение и функции редактора

Как было объяснено в предыдущей главе, с одной стороны, программы ST создаются в проекте, с другой – функциональные блоки и функции ST создаются в библиотеке.

Мы начнем с создания программы ST в контексте проекта.

Редактор ST – это текстовый редактор, обладающий рядом дополнительных функций. Команды и ключевые слова выделяются цветом.

```

1   PROGRAM _CYCLIC
2   TASK_SYS_CYCLIC();
3
4   (* Different base operations with variables *)
5   STC_BOOL_out := NOT STC_BOOL_in;
6
7   STC_USINT_out := STC_USINT_in + 1;
8   STC_ULINT_out := STC_ULINT_in - 1;
9
10  STC_REAL_out := STC_REAL_in + 1.001;
11  STC_LREAL_out := STC_LREAL_in / 2.5;
12
13  STC_WORD_out := NOT STC_WORD_in;
14  STC_DWORD_out := ROL( STC_DWORD_in, 1 );
15  STC_LWORD_out := SHL( STC_LWORD_in, 1 );
16
17  (* Function call with time and date functions from IEC-library *)
18  STC_DT_out := STC_DT_in + T#1s;
19  STC_TIME_out := STC_TIME_in + T#1s;
20
21  (* Function call of local function blocks *)
22  STC_MIN_INT_out := LOC_MIN_INT( STC_INT_in, 100 );
23  STC_MAX_REAL_out := LOC_MAX_REAL( 500, STC_REAL_in );
24
25  IF (STC_RS1_Q_out = 1) THEN
26    RS1.Rin := STC_BOOL_out;
27    RS1.Sin := 0;
28  ELSE
29    RS1.Rin := 0;
30    RS1.Sin := STC_BOOL_out;
31  END_IF;
32  RS1();
33  STC_RS1_Q_out := RS1.Qout;
34  STC_RS1_QN_out := RS1.QNout;
35
36
37  TASK_SYS_CYCLICEND();
38  END_PROGRAM
39

```

Рисунок 8: Редактор ST в APROL

#### Редактор содержит следующие функции и свойства:

- Различие между прописными и строчными буквами (чувствительность к регистру)
- Возможность объявления локальных функций и функциональных блоков
- Обзор всех переменных, используемых в коде
- Автоподстановка переменных
- Найти/заменить
- Отменить/вернуть
- Импорт кода ST
- Встроенная проверка синтаксиса в контекстном меню

# ST в разработке проектов

- Возможность использования функций и функциональных блоков из других библиотек APROL
- Идентификация соответствия пар скобок
- Развертывание и свертывание конструкций (очерчивание)
- Маркер измененной строки

## 4.2 Создание программ ST

Мы начнем с создания программы ST в проекте.

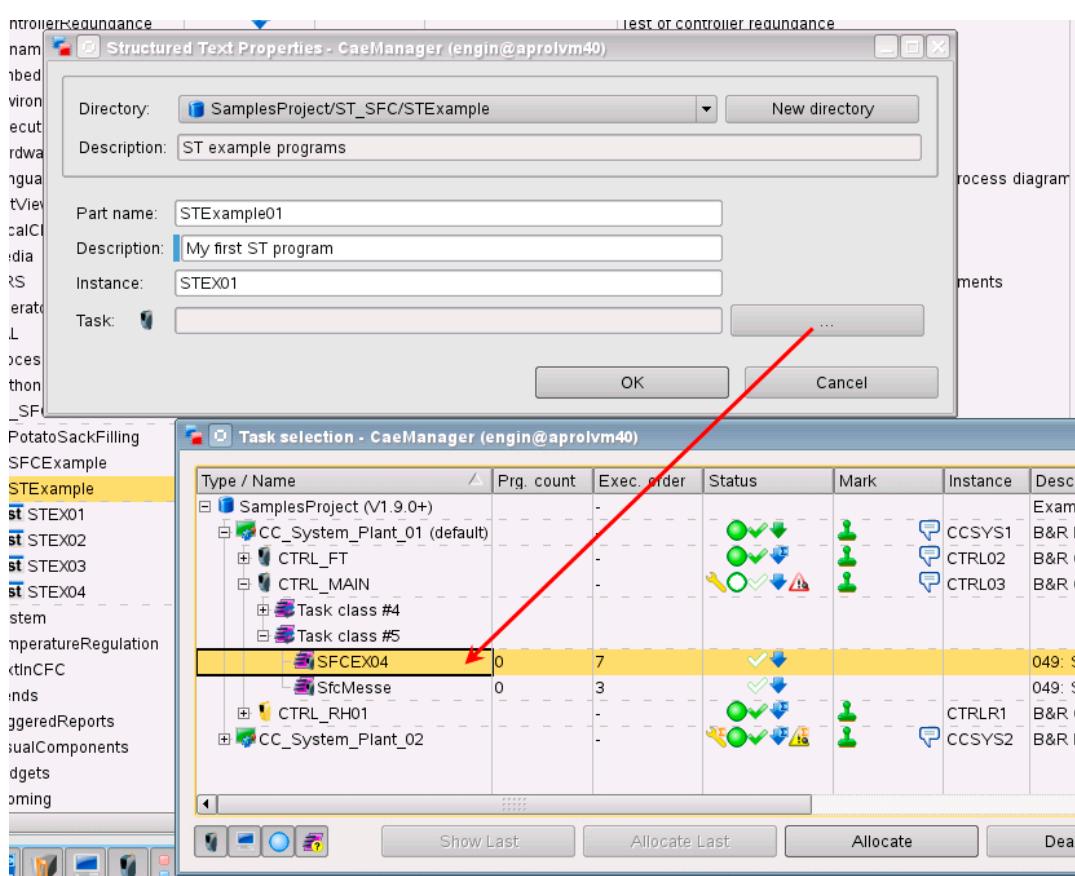


Рисунок 9: Создание программы ST и назначение задачи

Путь: CaeManager / CAE project / File / New / ST (Structured Text)



Part name (Имя)

<имя программы>

Instance (Экземпляр)

<имя экземпляра>

Task (Задача)

Выберите задачу при помощи кнопки [...]

Подтвердите нажатием кнопки [OK]



Задачу необязательно назначать сразу же. Это можно сделать позже. Однако программа ST не может быть активирована без назначения задачи.

Помимо этого, для каждой программы ST необходимо создать отдельную задачу контроллера. Этой задачей не может быть задача по умолчанию или задача, содержащая другие части проекта!

После создания программы ST можно выполнить компиляцию части проекта, сборку задачи и загрузку в аппаратную часть.

По завершении этих действий настройка программы ST считается завершенной.

#### 4.3 Работа с редактором ST

##### 4.3.1 Наиболее важные вкладки редактора ST:

###### **'Master data' (Основные данные)**

На этой вкладке отображаются основные параметры программы ST. Также здесь можно настроить имя экземпляра этой части проекта.



Рисунок 10: Основные данные в редакторе ST

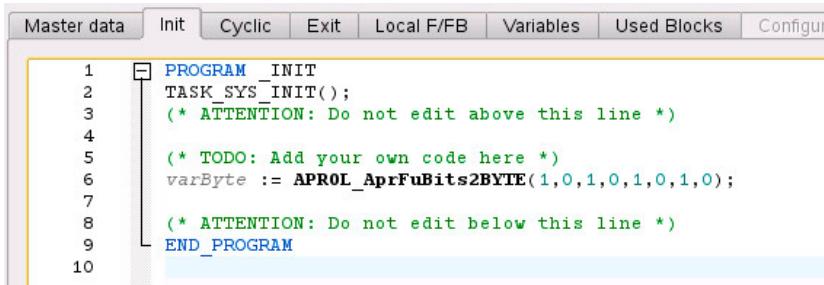
###### **Процедуры инициализации и выхода**

Вкладка **'Init'** (**Инициализация**) содержит текстовый редактор. Он предназначен для редактирования кода инициализации с использованием языка программирования ST. Код выполняется в Init-UP задачи и может быть использован для инициализации функций управления и переменных.

Вкладка **'Exit'** (**Выход**) также содержит текстовый редактор. Он предназначен для редактирования кода деинициализации и также использует ST. Выходной код выполняется в Exit-UP задачи и может быть использован, например, для деинициализации функций управления и переменных.

Программный код, содержащийся на двух указанных вкладках, обрабатывается/выполняется однократно.

# ST в разработке проектов



Master data Init Cyclic Exit Local F/FB Variables Used Blocks Configuration

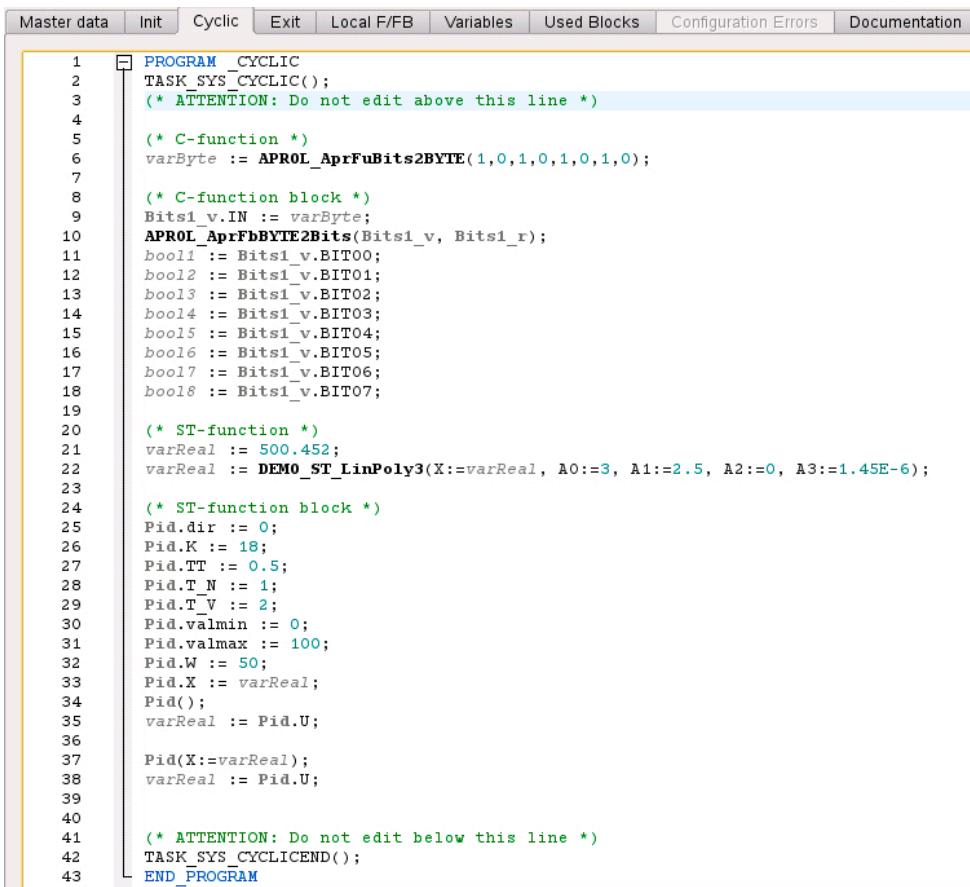
```
1 PROGRAM _INIT
2 TASK_SYS_INIT();
3 (* ATTENTION: Do not edit above this line *)
4
5 (* TODO: Add your own code here *)
6 varByte := APROL_AprFuBits2BYTE(1,0,1,0,1,0,1,0);
7
8 (* ATTENTION: Do not edit below this line *)
9 END_PROGRAM
10
```

Рисунок 11: Программный код на вкладке «Init»

## Циклическая часть кода

Циклический код программы ST находится на вкладке 'Cyclic' (Циклический). Код обрабатывается на контроллере циклически.

На этой вкладке можно запрограммировать все элементы, выражения и группы команд.



Master data Init Cyclic Exit Local F/FB Variables Used Blocks Configuration Errors Documentation

```
1 PROGRAM _CYCLIC
2 TASK_SYS_CYCLIC();
3 (* ATTENTION: Do not edit above this line *)
4
5 (* C-function *)
6 varByte := APROL_AprFuBits2BYTE(1,0,1,0,1,0,1,0);
7
8 (* C-function block *)
9 Bits1_v.IN := varByte;
10 APROL_AprFbBYTE2Bits(Bits1_v, Bits1_r);
11 bool1 := Bits1_v.BIT00;
12 bool2 := Bits1_v.BIT01;
13 bool3 := Bits1_v.BIT02;
14 bool4 := Bits1_v.BIT03;
15 bool5 := Bits1_v.BIT04;
16 bool6 := Bits1_v.BIT05;
17 bool7 := Bits1_v.BIT06;
18 bool8 := Bits1_v.BIT07;
19
20 (* ST-function *)
21 varReal := 500.452;
22 varReal := DEMO_ST_LinPoly3(X:=varReal, A0:=3, A1:=2.5, A2:=0, A3:=1.45E-6);
23
24 (* ST-function block *)
25 Pid.dir := 0;
26 Pid.K := 18;
27 Pid.TT := 0.5;
28 Pid.T_N := 1;
29 Pid.T_V := 2;
30 Pid.valmin := 0;
31 Pid.valmax := 100;
32 Pid.W := 50;
33 Pid.X := varReal;
34 Pid();
35 varReal := Pid.U;
36
37 Pid(X:=varReal);
38 varReal := Pid.U;
39
40
41 (* ATTENTION: Do not edit below this line *)
42 TASK_SYS_CYCLICEND();
43 END_PROGRAM
..
```

Рисунок 12: Циклическая часть программы ST

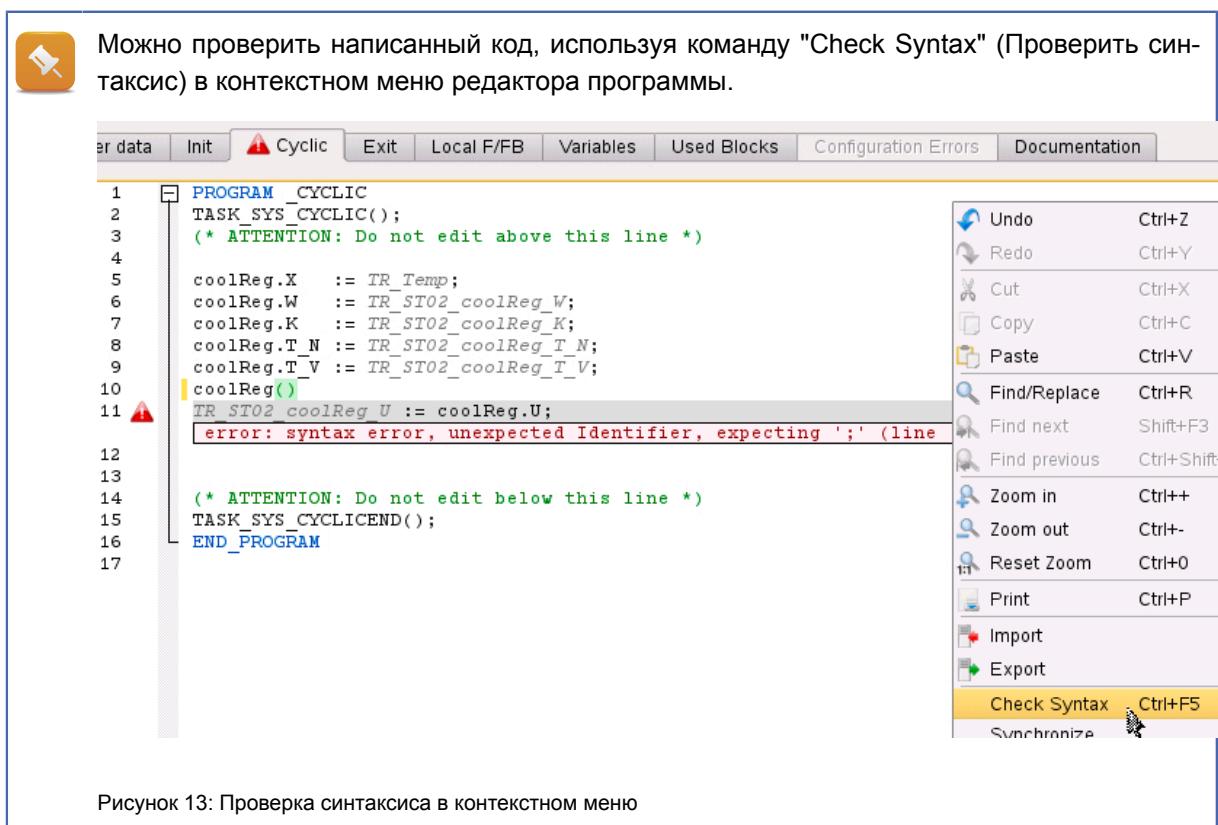


Рисунок 13: Проверка синтаксиса в контекстном меню

### Использование локальных и переменных проекта.

Все переменные, используемые в программном коде (на вкладках Init, Cyclic и Exit), создаются автоматически в ходе сохранения или проведения синхронизации вручную на вкладке 'Variables' (Переменные). Они приведены здесь вместе с их свойствами.

Master data											Init	Cyclic	Exit	Local F/FB	Variables	Used Blocks	Configuration Errors	Documentation
Project Variables																		
Name	Used	IEC Type	Live Value	Direction	I/O	Description												
CTRL_TR_DPS1_AT2311_CH1_H	✓	INT	---	→	→	049: Temperatur High Word												
CTRL_TR_DPS1_AT2311_CH1_L	✓	UINT	---	→	→	049: Temperatur Low Word												
CTRL_TR_DPS1_AT2311_CH2_H	✓	INT	---	→	→	049: Temperatur High Word												
CTRL_TR_DPS1_AT2311_CH2_L	✓	UINT	---	→	→	049: Temperatur Low Word												
CTRL_TR_PWL1_A04622_CH1	✓	REAL	---	→	→	049: Temperatur: 0 - 100%												
CTRL_TR_PWL1_A04622_CH2	✓	REAL	---	→	→	049: Temperatur: 0 - 100%												
TR_Cooler	✓	REAL	---	→	→	049: Temperatur: 0 - 100%												
TR_EnvTemp	✓	REAL	---	→	→	049: Temperatur °C												
TR_Heater	✓	REAL	---	→	→	049: Temperatur: 0 - 100%												
TR_Temp	✓	REAL	---	→	→	049: Temperatur °Kühlkörper °C												
Local Variables																		
Name	Used	Category	Pin Type/Data Type	Direction	Project Variable	Live Value	Const.	Rem.	RR	Default Value	Trans.	Description						
CoolerRunning	✓	IEC type	BOOL	→		---	✗	✗			✗							
Motor33	✓	IEC type	BOOL	→		---	✗	✗			✗							
NoValve	✓	IEC type	BOOL	→		---	✗	✗		False (0)	✗							
StepVariable	✓	IEC type	INT	→		---	✗	✗		1	✗							

Рисунок 14: Список переменных в редакторе ST

## ST в разработке проектов

Если была создана программа с множеством локальных переменных, некоторые из которых планируется сделать переменными проекта (т. е. коннекторами), тогда имя переменной проекта необходимо ввести в соответствующем столбце "Project variable" (Переменная проекта). Таким образом будет организована связь с переменной проекта и синхронизация значения.



При необходимости добавления аппаратных переменных, таких как дискретные входы и выходы, в код ST необходимо добавить в конец имени переменной .ha.

```
DigitalOutput1.ha := DigitalInput1.ha AND (AnalogInput3.ha > 10.0);
```



Синхронизацию вручную можно выполнить в любой момент времени при помощи пункта контекстного меню '**'Synchronize'** (Синхронизировать). При этом все вкладки редактора синхронизируются между собой, добавляются новые списки переменных или выполняется обновление статуса использования (см. зеленую галочку во втором столбце списка переменных).

```
running := 0;
tarten: Läuft erst bei 4.5V an
erRunning = 0) THEN
_PWL1_A04622_CH1.ha := 45;
unning := 1;
tung > 0 => Ausgangsspannung =
_PWL1_A04622_CH1.ha := 20 + (TR_Heating * 0.5);
ausgang abschalten bei Übertemperatur
> 75.0) THEN
_PWL1_A04622_CH2.ha := 0;
_PWL1_A04622_CH2.ha := TR_Heating;
3 + NoValve AND StepVariable;
N: Do not edit below this line *)
CLICEND();
```

Рисунок 15: Синхронизация в контекстном меню

**Задача: проверка уровня жидкости**

Необходимо отслеживать уровень жидкости в емкости в трех областях: низкий, высокий и нормальный.

Используйте аппаратные выходы для обозначения каждого из уровней – низкого, нормального, высокого.

Уровень жидкости считывается в виде аналогового значения (0–32767), затем выполняется внутреннее преобразование в процентное значение (0–100 %).

Если уровень жидкости падает ниже 25 %, должен быть запущен предупреждающий звуковой сигнал.

Решите эту задачу, используя оператор CASE в программе ST.



Рисунок 16: Мониторинг уровня жидкости в емкости

**Объявление:**

```

VAR
    aiLevel : INT;
    PercentLevel : UINT;
    doLow : BOOL;
    doOk : BOOL;
    doHigh : BOOL;
    doAlarm : BOOL;
END_VAR

```

Таблица 3: Пример объявления переменных

#### 4.3.3 Локальные функции и функциональные блоки

Из этого раздела вы узнаете, как создавать в редакторе ST функции и функциональные блоки, а также как использовать их в коде. Эти блоки могут быть использованы только в программах ST.

Программирование и создание экземпляров функций и функциональных блоков осуществляется на вкладке 'Local F/FB' (Локальная функция/Функц. блок).

# ST в разработке проектов

The screenshot shows the ST editor interface with three main tabs: Declaration, Instantiation, and Definition.

**Declaration:**

```
1 FUNCTION LOC_MIN_INT : INT
2   VAR_INPUT
3     IN1 : INT;
4     IN2 : INT;
5   END_VAR
6 END_FUNCTION

7 FUNCTION LOC_MAX_REAL : REAL
8   VAR_INPUT
9     IN1 : REAL;
10    IN2 : REAL;
11  END_VAR
12 END_FUNCTION
```

**Instantiation:**

```
1 VAR RETAIN
2   RS1: LOC_FLIPFLOP;
3 END_VAR

4 VAR CONSTANT
5
6 END_VAR

7 VAR
8
9 END_VAR
```

**Definition:**

```
2 FUNCTION LOC_MIN_INT
3   IF IN1 < IN2 THEN
4     LOC_MIN_INT := IN1;
5   ELSE
6     LOC_MIN_INT := IN2;
7   END_IF
8 END_FUNCTION

9
10 FUNCTION LOC_MAX_REAL
11   IF IN1 > IN2 THEN
12     LOC_MAX_REAL := IN1;
13   ELSE
14     LOC_MAX_REAL := IN2;
15   END_IF
16 END_FUNCTION

17 FUNCTION_BLOCK LOC_FLIPFLOP
18   IF Rin=1 THEN
19     Qout := 0;
20   ELSIF Sinv=1 THEN
21     Qout := 1;
22   END_IF
23   QOut := NOT Qout;
24 END_FUNCTION_BLOCK
```

Рисунок 17: Примеры локальных функций и функциональных блоков

**Они создаются следующим образом:**

- 1) Сначала осуществляется объявление функции или функционального блока в левой верхней области, т. е. здесь определяются их переменные.

```
FUNCTION LOC_MINIMUM_INT : INT
  VAR_INPUT
    IN1 : INT;
    IN2 : INT;
  END_VAR
END_FUNCTION

FUNCTION_BLOCK LOC_FLIPFLOP
  VAR_INPUT
    R_IN : BOOL;
    S_IN : BOOL;
  END_VAR
  VAR_OUTPUT
    Q_OUT : BOOL;
    QN_OUT : BOOL;
```

```

    END_VAR
END_FUNCTION_BLOCK

```

- 2) После этого определяется цель (алгоритм) функции (блока). В нашем примере мы создаем функцию "LOC\_MINIMUM\_INT", возвращающую наименьшее из значений двух входных переменных, и функциональный блок типа «триггер» "LOC\_FLIPFLOP".

```

FUNCTION LOC_MINIMUM_INT
    IF IN1 < IN2 THEN
        LOC_MINIMUM_INT := IN1;
    ELSE
        LOC_MINIMUM_INT := IN2;
    END_IF
END_FUNCTION

```

```

FUNCTION_BLOCK LOC_FLIPFLOP
    IF R_IN = 1 THEN
        Q_OUT := 0;
    ELSIF S_IN = 1 THEN
        Q_OUT := 1;
    END_IF
    QN_OUT := NOT Q_OUT;
END_FUNCTION_BLOCK

```

- 3) Перед использованием блока в циклическом коде необходимо создать экземпляр и объявить его как волатильный или реманентный. В процессе создания экземпляра определяется его уникальное имя, вызывающее функциональный блок. Может быть создано несколько экземпляров функционального блока одного типа.

```

VAR RETAIN
    RS1 : LOC_FLIPFLOP
END_VAR

```



Переменная RS1 предназначена для «адресации» созданного функционального блока. Мы определяем ее как «сохраняющуюся» (retain), подразумевая, что все выводы являются реманентными.

Функции не требуют создания экземпляров и могут быть использованы непосредственно в программном коде.

- 4) Теперь мы можем использовать созданные функции и функциональные блоки в программном коде (вкладки Init, Cyclic и Exit) следующим образом:

```

AnalogValue_Min := LOC_MINIMUM_INT( AnalogVal_01, AnalogVal_02 );

//Declare inputs
RS1.R_IN := DigitalValue_10;
RS1.S_IN := DigitalValue_11;

RS1();      //Call function block

DigitalValue_12 := RS1.Q_OUT;      //Read-back output value

```

#### 4.3.4 Вызов существующих блоков САЕ в ST

Поскольку блоки библиотеки САЕ содержат множество функций, необходимость в программировании таких же «с нуля» отсутствует; их можно использовать непосредственно в программе ST.

## ST в разработке проектов

Все функциональные блоки, используемые в контексте программы ST, перечислены на вкладке 'Used Blocks' (Используемые блоки).

Функциональные блоки можно переместить в эту таблицу из контекста окна навигации посредством простого перетаскивания.

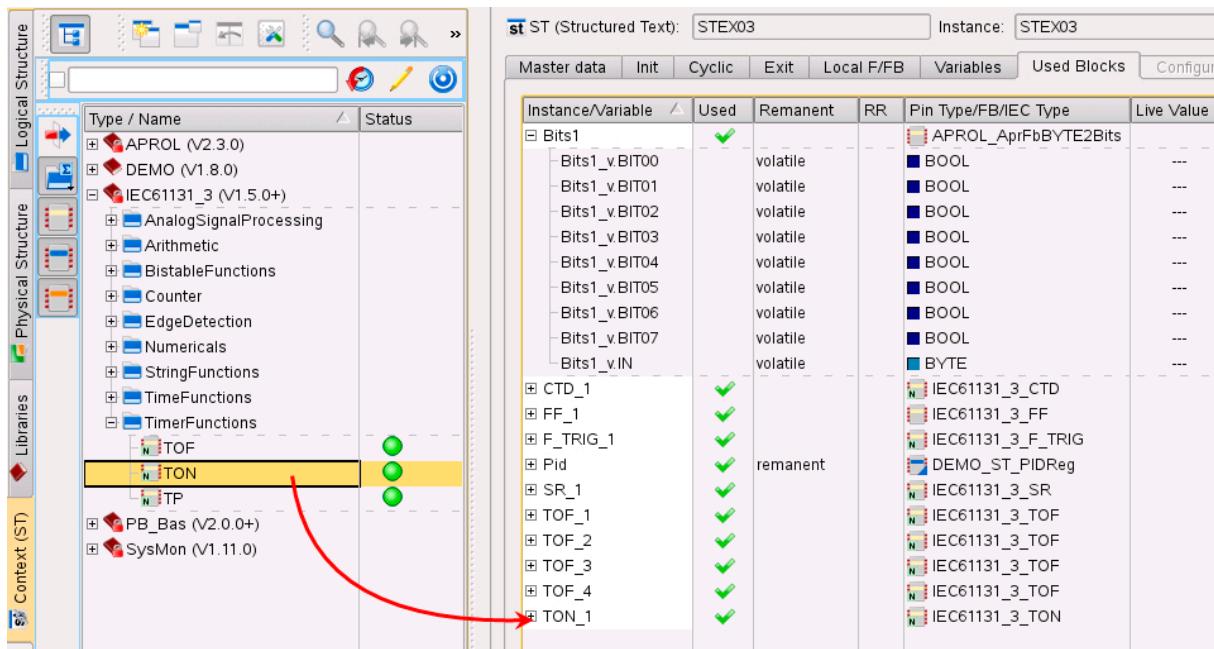


Рисунок 18: Вид 'Used Blocks' CAE

Для каждого функционального блока приводится таблица с его вводами и выводами. Вид имеет древовидную структуру, в понятной форме отображающую отдельные функциональные блоки и их вводы и выводы. Каждый функциональный блок представлен корневым элементом, а соответствующие вводы и выводы отображаются как подчиненные элементы.

### Столбец

Instance / Variable (Экземпляр / Переменная)	Контекст функционального блока: имя экземпляра, при помощи которого происходит вызов используемого функционального блока. Имена экземпляров можно редактировать. Контекст ввода/вывода: имя переменной ввода или вывода.
Status of usage (Состояние использования)	Показывает, используется ли блок в контексте этой программы после проведения синхронизации.
Remanent (Энергонезависимый)	Показывает, является ли ввод/вывод (блок C) или блок (блок ST/SFC) реманентным.
Pin type / FB / IEC type (Тип контакта / FB / IEC)	<b>Контекст ввода/вывода:</b> тип IEC или тип контакта для ввода/вывода. <b>Контекст функционального блока:</b> имя функционального блока в том виде, в котором оно должно использоваться в коде.
Live value (Фактическое значение)	Значение переменной процесса в режиме отладки.
I/O constant (Константа ввода-вывода)	Здесь можно ввести значение постоянной ввода/вывода, отличное от значения по умолчанию из библиотеки.

Столбец	
I/O (Ввод/вывод)	Контекст ввода/вывода: указывает, является ввод/вывод вводом или выводом.
Block description (Описание блока)	Описание из библиотеки CAE.
Instance description (Описание экземпляра)	Описание определенного экземпляра, которое может быть введено пользователем.

После размещения функционального блока в таблице его можно использоваться в коде ST также путем простого перетаскивания из неё; входа функционального блока при этом получат свои необходимые значения по умолчанию.



**Функции** из имеющихся библиотек CAE нельзя разместить в таблице. Они размещаются непосредственно прямо в код ST в форме <имя библиотеки>\_<имя блока>.

```

Master data Init Cyclic Exit Local F/FB Variables Used Blocks Configuration
1 PROGRAM _CYCLIC
2 TASK SYS_CYCLIC();
3 (* ATTENTION: Do not edit above this line *)
4
5 (* C-function *)
6 varByte := APROL_AprFuBits2BYTE(1,0,1,0,1,0,1,0);
7
8 (* C-function block *)
9 Bits1_v.IN := varByte;
10
11 APROL_AprFbBYTE2Bits(Bits1_v, Bits1_r);
12
13 bool1 := Bits1_v.BIT00;
14 bool2 := Bits1_v.BIT01;
15 bool3 := Bits1_v.BIT02;
16 bool4 := Bits1_v.BIT03;
17 bool5 := Bits1_v.BIT04;
18 bool6 := Bits1_v.BIT05;
19 bool7 := Bits1_v.BIT06;
20 bool8 := Bits1_v.BIT07;
21
22

```

Рисунок 19: Вызов функций и блоков CAE

### Задача: использовать функциональный блок из имеющейся библиотеки

Дополните предыдущее упражнение «Проверка уровня жидкости» блоком таймера TON (из библиотеки IEC61131\_3 для CAE APROL).

Можно, например, указать время ожидания перед срабатыванием предупреждающего сигнала в случае, если уровень содержимого в емкости упал ниже 25 %.

## 4.4 Отладка программ ST

### 4.4.1 Значения переменных для отладки

Программа ST переключается в режим отладки для просмотра и проверки значений переменных.

## ST в разработке проектов



Рисунок 20: Включение отладки в реальном времени

Режим отладки можно включить и отключить на панели инструментов CaeManager. После включения столбец "Live value" (Значение переменной для отладки) в обзоре переменных отображает текущее значение переменной.

Name	Used	IEC Type	Live Value	Direction	I/O	Description
C01_ST09_CH03_DO	✓	BOOL	1	→	→	Control variable of the Oil pump for the centrifugal pump
C01_ST09_CH04_DO	✓	BOOL	0	→	→	Control variable of the Twonveyor direction vessel
C01_ST09_CH05_DO	✓	BOOL	0	→	→	Control variable of the Twirection storage conveyor
C01_ST09_CH06_DO	✓	BOOL	0	→	→	Control variable of the Chcrystal product to storage
C01_ST09_CH07_DO	✓	BOOL	1	→	→	Control variable of the Stirrer
C01_ST09_CH08_DO	✓	BOOL	0	→	→	Control variable of the Outflow pump
C01_ST09_CH09_DO	✓	BOOL	1	→	→	Control variable of the Recirculation pump
DC9817_Sim	✓	REAL	4910	→	→	DC9817 Simulation Mode
EI9807_Sim	✓	REAL	55	→	→	EI9807 Simulation Mode
EQ03_density_IN	✓	REAL	3608	→	→	density of the fluid product
FC9812_Sim	✓	REAL	44.552921295166	→	→	FC9812 Simulation Mode
FQ9809_Sim	✓	REAL	0	→	→	Simulation Mode
LA9809_Sim	✓	BOOL	0	→	→	Simulation Mode
LC9809_Alarm2	✓	BOOL	0	→	→	LC9809 high high alarm

Name	Used	Category	Pin Type/Data Type	Direction	Project Variable	Live Value	Const.	Rem.	RR	Default Value
aus	✓	IEC type	BOOL	→		0	✓	✗		False (0)
bits	✓	IEC type	DWORD	→		2	✗	✗		
bits2	✓	IEC type	DWORD	→		2	✗	✗		
ein	✓	IEC type	BOOL	→		1	✓	✗		True (1)

Рисунок 21: Список переменных со значениями для отладки



Столбец "Live value" также актуален для списка используемых блоков, он отображает текущее значение вводов и выводов функциональных блоков.

### 4.4.2 Отслеживание переменных в ControllerManager

Другим способом отладки программы ST является использование отслеживания переменных (Watch) в ControllerManager. С помощью окна Watch можно отобразить все переменные программы ST вместе с их значениями, а также при необходимости изменить значения (т. е. принудительно указать их).

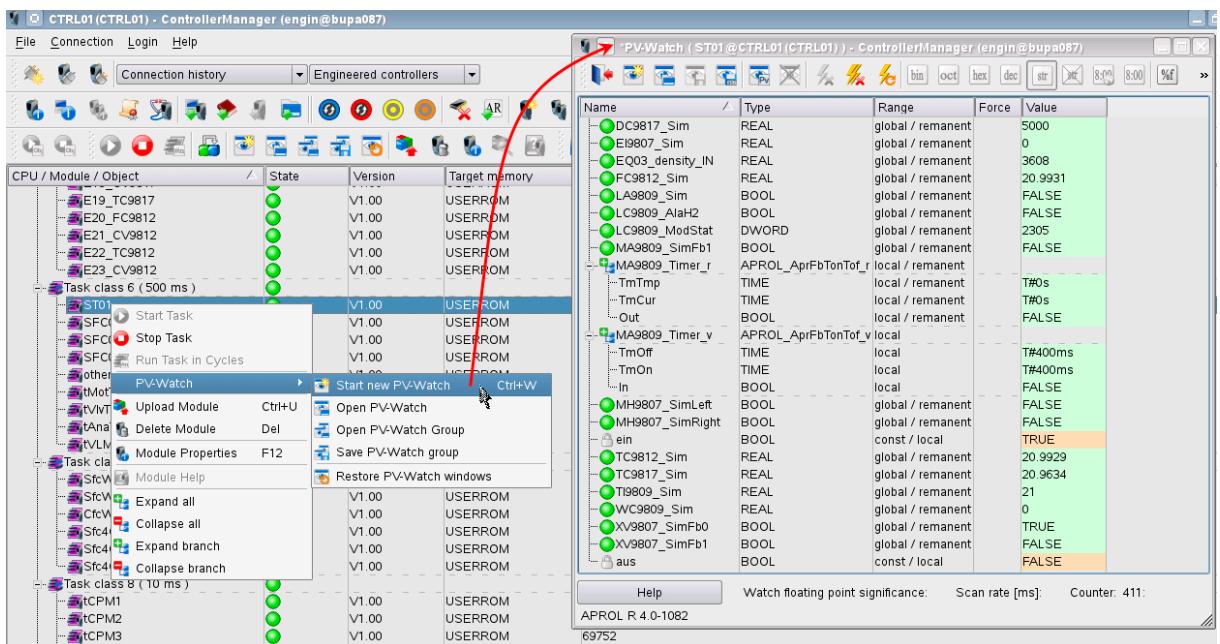


Рисунок 22: Окно отслеживания переменных (Watch) в ControllerManager

Для доступа к окну Watch откройте ControllerManager. Выберите задачу программы ST, а затем выберите в контекстном меню пункт **Create New PV Watch** (Создать новое отслеживание переменных процесса). Можно переместить в это окно для просмотра значений отдельные переменные или все сразу. Здесь можно принудительно указать значения переменных в зависимости от типа данных.

# SFC в разработке проектов

## 5 SFC в разработке проектов

### 5.1 Введение и функции редактора

При создании последовательной функциональной схемы (SFC) существует разделение на программы SFC в проекте и функциональные блоки SFC в библиотеке.

В программу SFC вставляются шаги, переходы, прыжки и действия.

**Редактор SFC в APROL можно вызывать различными способами:**

- Панель инструментов: Создать
- Из контекстного меню программы SFC, шага или перехода

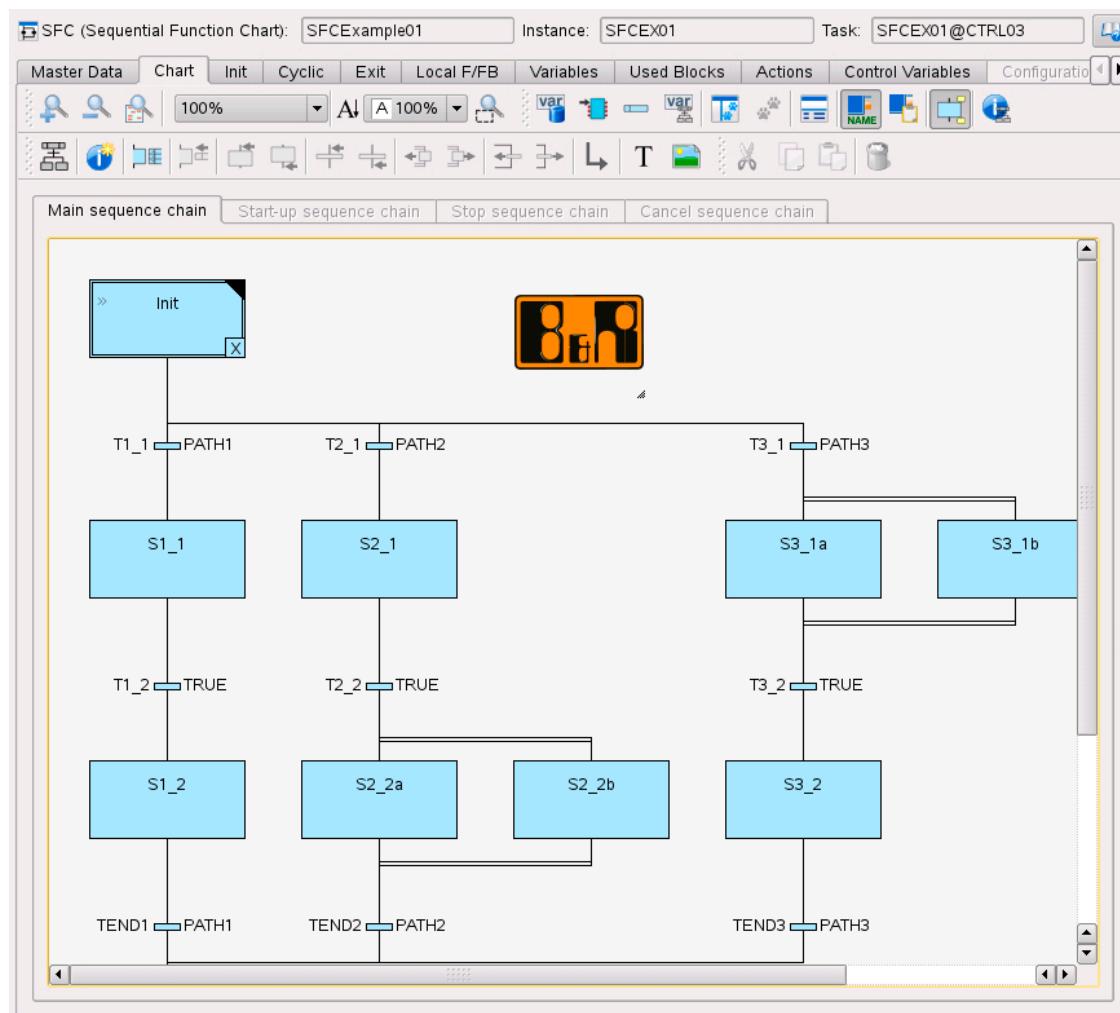


Рисунок 23: Редактор SFC

Добавление объектов выполняется при помощи строки меню или контекстных меню. Указание свойств объекта выполняется в соответствующем диалоговом окне. Открыть это окно можно, дважды щелкнув на отмеченном элементе.

Для увеличения/уменьшения масштаба в редакторе SFC можно использовать панель инструментов или сочетания клавиш.



Редактор SFC, как и редактор ST, содержит функции автоподстановки, разрешает использование локальных функций и функциональных блоков, различает прописные и строчные буквы.

## 5.2 Создание программы SFC

Далее объясняется создание программы SFC в проекте

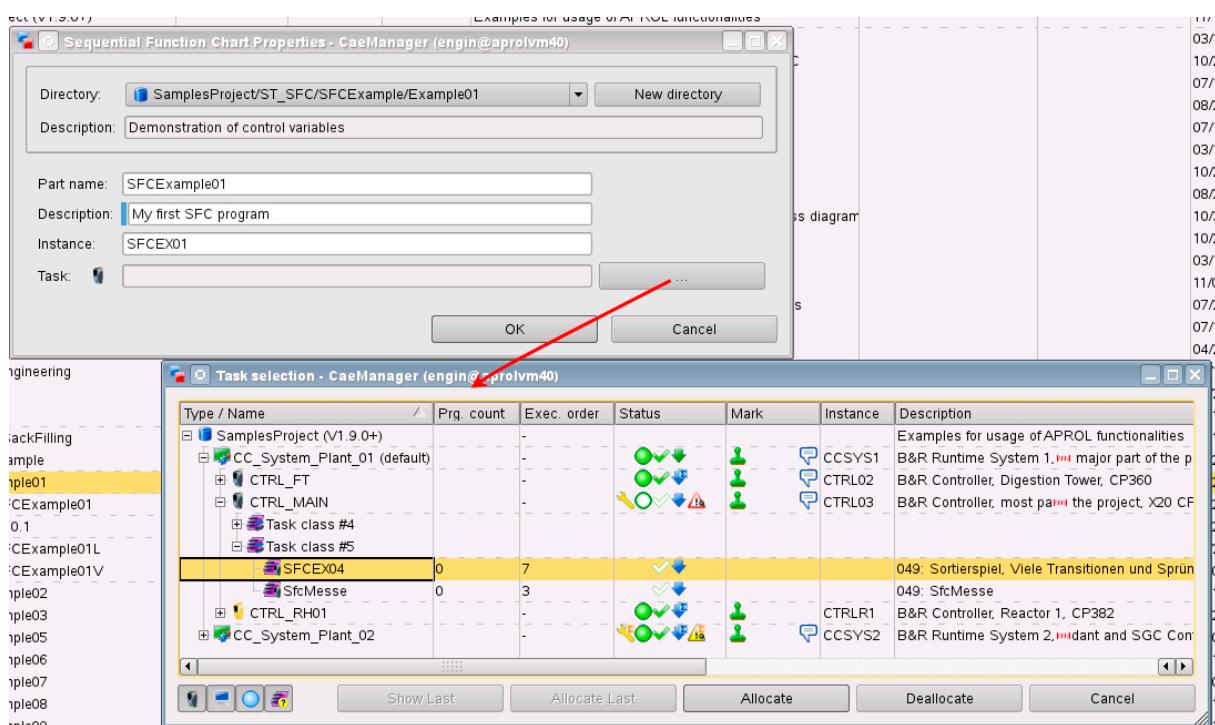


Рисунок 24: Создание программы SFC и назначение задачи

Путь: CaeManager / CAE Project / File / New / SFC (Sequential Function Chart)



Part name (Имя)

<имя программы>

Instance (Экземпляр)

<имя экземпляра>

Task (Задача)

Выберите задачу при помощи кнопки [...]

Подтвердите нажатием кнопки [OK]



Задачу необходимо назначать сразу же. Это можно сделать позже; однако программа SFC не может быть активирована без назначения задачи.

Помимо этого, для каждой программы SFC необходимо создать отдельную задачу контроллера. Этой задачей не может быть задача по умолчанию или задача, содержащая другие части проекта!



Дополнительная информация о назначении задаче приведена в разделе [Какими особыми функциями обладает SFC в APROL?](#)

После создания программы SFC можно выполнить компиляцию части проекта, создание задачи и загрузку в аппаратную часть.

По завершении этих действий настройка программы SFC считается завершенной.

## 5.3 Работа с редактором SFC

### 5.3.1 Вкладки редактора

Как вы могли заметить при создании или открытии программы SFC, многие вкладки в редакторе идентичны вкладкам редактора ST.

**Такой список приведен ниже. Для получения дополнительной информации см.[Наиболее важные вкладки редактора ST:](#)**

- Init (Инициализация)
- Exit (Выход)
- Local F/FB (Локальные ф-ция/функция блок)
- Variables (Переменные)
- Used Blocks (Используемые Блоки)
- Configuration Errors (Ошибки конфигурации)
- Documentation (Документация)

**Дополнительные вкладки и их функции объяснены ниже.**

**'Master data' (Основные данные):**

на данной вкладке указываются основные параметры программы SFC.

The screenshot shows the 'Master Data' tab of the SFC editor interface. The tab bar includes: Master Data, Chart, Init, Cyclic, Exit, Local F/FB, Variables, Used Blocks, Actions, Control Variables, Configuration, and Help. The 'Master Data' tab is active. The form contains the following fields:

- Name: SFCEExample01
- Description: SFC example 1: control variables
- Instance: SFCEX01
- Task: SFCEX01@CTRL03
- Control Options:
  - Controllable by SFCViewer
- Remanence:
  - State and control variables remanent
- Step / Transition labeling (default):
  - Name
  - Short description
- Object size:
  - Object width: 1.0
  - Object height: 1.0
- Grid setting:
  - Horizontal spacing: 1.0
  - Vertical spacing: 1.0

Рисунок 25: Основные данные программы SFC

При активации параметра '**Controllable by SFCViewer**' (Управляется через SFCViewer) оператор может вручную управлять обработкой программы SFC (например, остановить программу, принудительно запустить определенные шаги и действия) в SFCViewer.

При активации параметра '**Status and control variables remanent**' (Переменные состояния и управляющие переменные реманентны) все переменные состояния шагов, переходов и действий, а также управляющие переменные SFCViewer (при включенном параметре 'Controllable by SFCViewer'), создаваемые автоматически в части проекта, будут являться реманентными.

Ниже можно установить различные параметры компоновки/отображения объектов SFC.

### Вкладка 'Chart' (Схема)

Наиболее важная вкладка. Здесь происходит графическая компоновка программы SFC.

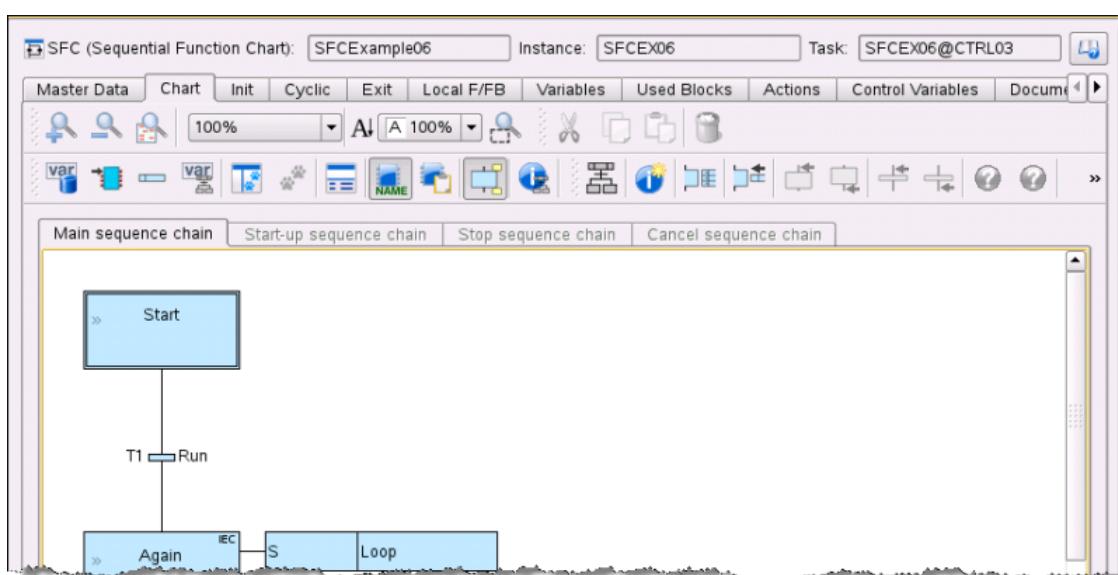


Рисунок 26: Схема SFC

#### **Множественный выбор с использованием клавиатуры и мыши:**

в редакторе SFC и SFCViewer можно выбрать несколько шагов или переходов, удерживая клавиши [Ctrl] и/или [Shift], а затем перейти к свойствам выбранных элементов при помощи пункта '**Properties**' (Свойства). Преимущество этого способа выделения состоит в том, что общие параметры могут быть изменены одним действием.

Используя клавиши **{[Ctrl]} и/или {[Shift]}**, можно также выбрать несколько начальных точек последовательности и переместить их при помощи пункта контекстного меню '**Increase/decrease alternative path priority**' (Повысить/понизить приоритет альтернативного пути) в редакторе SFC. Для этого необходимо выбрать первый переход в каждом из существующих путей.

#### **Размещение изображений в SFC:**

для повышения читаемости программы и улучшения внешнего вида можно разместить в любом месте SFC изображения, содержащиеся в контейнере изображений. После этого размер изображений при необходимости может быть изменен. Размещение изображений под слоем объектов SFC с тем, чтобы изображения не закрывали объекты SFC, доступно установкой соответствующего параметра в инспекторе объектов.

# SFC в разработке проектов



Также на схеме можно разместить «произвольный текст» (Free text), подробно описывающий один или несколько шагов.

## Полученный код SFC

Код SFC, полученный из графической конфигурации, автоматически появляется на вкладке 'Cyclic' (Циклически).

```
Master Data Chart Init Cyclic Exit Local F/FB Variables Used Blocks Act.  
1 PROGRAM CYCLIC  
2 (* ATTENTION: Do not edit above this line *)  
3  
4 (* AUTOMATICALLY GENERATED AT: 2013.03.18 10:08:28 *)  
5  
6 INITIAL_STEP Init:  
7 (* @LANGUAGE := 'st' *)  
8 IdleCnt := IdleCnt + 1;  
9 IF IdleCnt > 10 THEN  
10 ....  
11 IdleCnt := 1;  
12 END_IF  
13 END_STEP  
14 (* @SFCCMAXTIME := '' *)  
15 (* @SFCCMINTIME := '' *)  
16 EXIT_ACTION  
17 (* @LANGUAGE := 'st' *)  
18 IdleCnt := -1;  
19 END_ACTION  
20  
21 TRANSITION T1_1 FROM Init TO S1_1 :=  
22 (* @LANGUAGE := 'st' *)  
23 PATH  
24 END_TRANSITION  
25  
26 STEP S1_1:  
27 (* @LANGUAGE := 'st' *)  
28 END_STEP  
29 (* @SFCCMAXTIME := 'T#10s' *)  
30 (* @SFCCMINTIME := 'T#5s' *)  
31 TRANSITION T1_2 FROM S1_1 TO S1_2 :=  
32 (* @LANGUAGE := 'st' *)  
33 TRUE  
34 END_TRANSITION  
35
```

Рисунок 27: Отображение кода программы SFC

Циклическая часть программного кода предоставляется только для чтения. Настройка осуществляется только графически на вкладке 'Chart' (Схема). Полученный код SFC необходим для генерации кода и отображается для того, чтобы внести в программу большую ясность.

## Объявление действий

Локальные действия могут быть определены только на языке ST на вкладке 'Actions' (Действия).

Name	State	Used	Description
AC_KlappenAuf		✓	049:
AC_KlappenZu		✓	049:

Definition

```
1 ACTION AC_KlappenAuf:  
2 KartSim_K5 := WK.Waage1;  
3 KartSim_K6 := WK.Waage2;  
4 KartSim_K7 := WK.Waage3;  
5 KartSim_K8 := WK.Waage4;  
6 KartSim_K9 := WK.Waage5;  
7  
8 END_ACTION
```

Рисунок 28: Вкладка 'Actions' программы SFC



Информация об использовании действий приведена в разделе [Действия в шагах IEC](#).

## Управляющие переменные SFC

Переменные, обеспечивающие дополнительные функциональные возможности, при необходимости можно объявить на вкладке 'Control variables' (Управляющие переменные).

Name	Used	Pin Type/Data	Direction	Project Variables	Live	Const.	Rem.	RR	Default Value	Description
SFCActionError	✗	■ BOOL	→	...	✗	✗	✗			TRUE if triggering of an action violates the actions as defined by TRUE if an error has been encountered during the self-diagnosis.
SFCCheckError	✗	■ BOOL	→	...	✗	✗	✗			Sets the mode for the self-diagnostic.
SFCCheckMode	✗	■ USINT	→	...	✗	✗	✗			Displays the name of the current step.
SFCCurrentStep	✗	■ SSTRING	→	...	✗	✗	✗			Enable recording of violations of time limits.
SFCEnableLimit	✗	■ BOOL	→	...	✗	✗	✗			Indicates violations of time limits.
SFCError	✗	■ BOOL	→	...	✗	✗	✗			Displays the name of the function block where time violations occur.
SFCErrorPOU	✗	■ SSTRING	→	...	✗	✗	✗			Displays the name of the step inducing time violations.
SFCErrorStep	✗	■ SSTRING	→	...	✗	✗	✗			TRUE if an action in the SFC has the force status ACTIVE.
SFCForceActionActive	✗	■ BOOL	→	...	✗	✗	✗			TRUE if an action in the SFC has the force status INACTIVE.
SFCForceActionInactive	✗	■ BOOL	→	...	✗	✗	✗			TRUE if an action that is attached to the SFC has the force status ACTIVE.
SFCForceActionStepInactive	✗	■ BOOL	→	...	✗	✗	✗			TRUE if a step in the SFC has the force status ACTIVE.
SFCForceStepActive	✗	■ BOOL	→	...	✗	✗	✗			TRUE if a step in the SFC has the force status INACTIVE.
SFCForceStepInactive	✗	■ BOOL	→	...	✗	✗	✗			TRUE if a transition condition in the SFC has the force status ACTIVE.
SFCForceTransActive	✗	■ BOOL	→	...	✗	✗	✗			TRUE if a transition condition in the SFC has the force status INACTIVE.
SFCForceTransInactive	✗	■ BOOL	→	...	✗	✗	✗			Resets the SFC program back to the initial step.
SFCInit	✗	■ BOOL	→	...	✗	✗	✓			Contains the ChronoLog identifier for the SFCLogging report.
SFCInstance	✗	■ SSTRING	→	...	✗	✗	✗			Enables logging for the SFC.
SFCLoggingMode	✗	■ USINT	→	...	✗	✗	✗			Pauses the execution of a function block / an SFC program.
SFCPause	✗	■ BOOL	→	...	✗	✗	✗			If the variable is set to TRUE, all of the variables which have an error
SFCQuitError	✗	■ BOOL	→	...	✗	✗	✗			Contains the SFC system variables of either second set, depending
SFCSystemVariables	✗	■ UDINT	→	...	✗	✗	✗			If TRUE, all transitions are evaluated as TRUE and the next step will
SFCTip	✗	■ BOOL	→	...	✗	✗	✗			If TRUE, a stepping with SFCTip is only performed if the SF
SFCTipConditional	✗	■ BOOL	→	...	✗	✗	✗			If TRUE, none of the step's actions are executed when the SFCTip is
SFCTipDisableAction	✗	■ BOOL	→	...	✗	✗	✗			If TRUE and SFCTip is used for stepping, forcing the evaluation of the
SFCTipDisableDuration	✗	■ BOOL	→	...	✗	✗	✗			If TRUE, the variables SFCTip and SFCTipMode have no effect on the
SFCTipLock	✗	■ BOOL	→	...	✗	✗	✗			If TRUE, all transition conditions are viewed as being dependent on
SFCTipMode	✗	■ BOOL	→	...	✗	✗	✗			This variable becomes TRUE when a transition is switched.
SFCTrans	✗	■ BOOL	→	...	✗	✗	✗			

Рисунок 29: Управляющие переменные SFC



Дополнительная информация об использовании управляемых переменных приведена в разделе [Использование управляемых переменных SFC](#)

### 5.3.2 Графическая настройка сети SFC

Базовая конфигурация (минимальная) программы SFC показана на следующем примере.

# SFC в разработке проектов

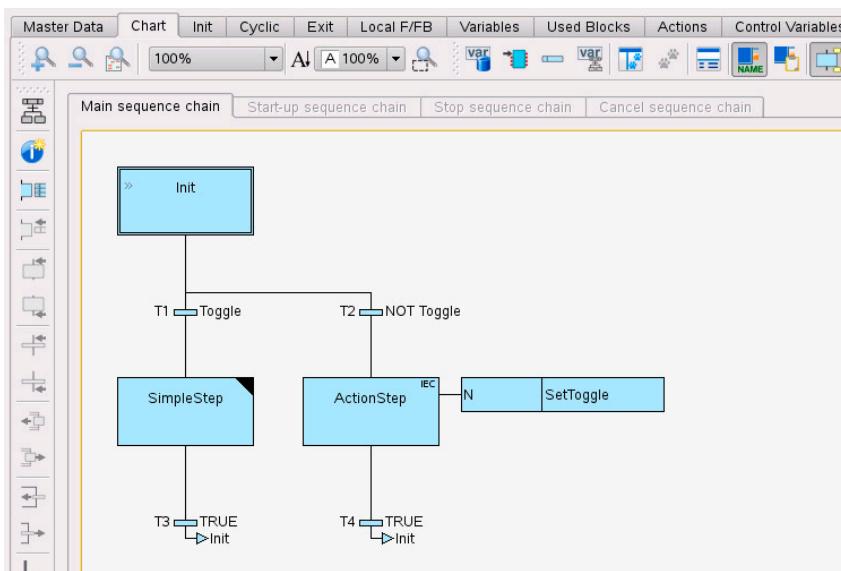


Рисунок 30: Пример схемы для программы SFC

Показанная ниже программа SFC (приведен пример завершенной конфигурации) выполняет переключение между двумя альтернативными путями.

На примере показано использование простых шагов, шагов IEC и действий.

В контексте примера необходимо выполнить следующие шаги:

Шаг	Разработка
1	Создание программы SFC
2	Настройка основных данных
3	Настройка схемы: Создание перехода/создание шага/создание прыжка
4	Создание действия Назначение действия/настройка шага

## Шаги 1 и 2: создание программы SFC и настройка основных данных

Инструкции по созданию программы SFC и настройке основных данных уже были приведены выше, в разделе [Создание программы SFC](#).

## Шаг 3: графическая настройка сети SFC

Начнем с редактирования существующих переходов после шага инициализации.

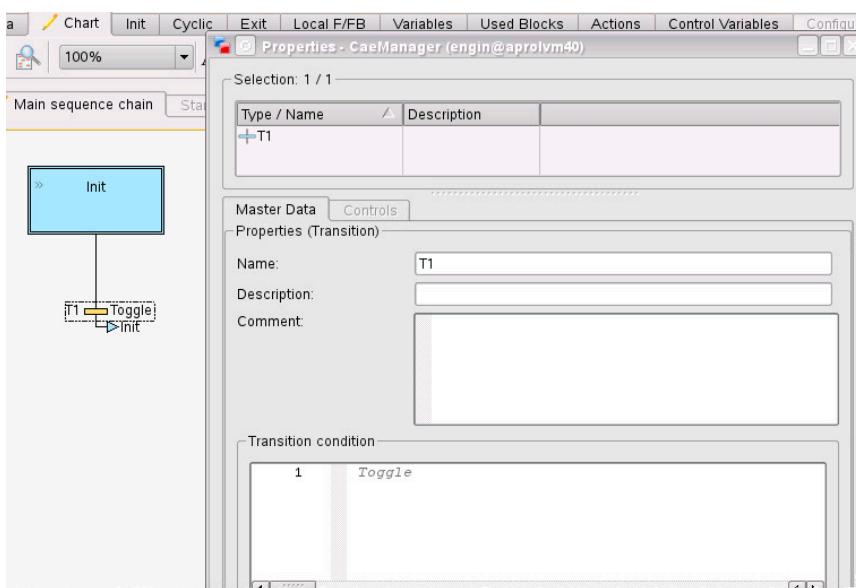


Рисунок 31: Диалоговое окно «Свойства» для переходов

**Переход '\_T1' / пункт контекстного меню 'Properties' (Свойства) или двойной щелчок:**

Name (Имя)	T1
Transition condition (Условие перехода)	Toggle
Кнопка [Close] («Закрыть»)	



Имя, описание или комментарий можно настроить для каждого объекта в SFC.

После перехода вставим простой шаг. Для простого шага необходимо настроить действия 'Entry', 'Cyclic' и 'Exit'.



Следует отличать входное и выходное действия от входного и выходного кода части проекта!

Входное действие выполняется при входе (переходе в активное состояние) назначенного **шага**, а выходное действие выполняется однократно при выходе из шага.

В отличие от них коды, расположенные на вкладках 'Init' и 'Exit' программы, выполняются в процессе инициализации и деинициализации программы SFC целиком на **контроллере**.

Снизить нагрузку на память контроллера можно использованием шагов типа **simple** (простой) (по сравнению с шагом типа IEC).

Теперь выделим переход 'T1' и выберем пункт контекстного меню 'Create step behind' (Создать шаг после перехода).

## SFC в разработке проектов

Шаг '\_S1' / пункт контекстного меню  
'Properties' (Свойства):

Name (Имя)	SimpleStep
Min. retention time (Мин. время удержания)	T#5s
Cyclic action (Циклическое действие)	Toggle := FALSE;
Следующий переход '_T1' / пункт контекстного меню 'Properties' (Свойства)	
Name (Имя)	T3
Transition condition (Условие перехода)	TRUE
Кнопка [Close] («Закрыть»)	

После этого мы продолжим с созданием альтернативного пути.

Переход 'T1', выберите пункт контекстного меню 'Insert alternative path right' (Вставить альтернативный путь справа)

Переход '\_T2' / пункт контекстного меню 'Properties' (Свойства):

Name (Имя)	T2
Transition condition (Условие перехода)	NOT Toggle
Кнопка [Close] («Закрыть»)	

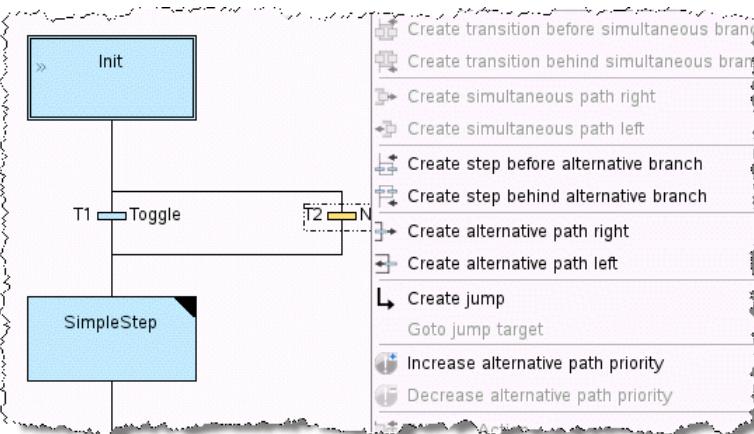


Рисунок 32: Следующий шаг: Добавление прыжка



Следует обратить внимание на то, что при создании прыжка в качестве цели автоматически указывается предыдущий шаг. Шаг, на который прыжок переведет, при необходимости можно изменить в диалоговом окне свойств.

В нашем примере в качестве цели выбран шаг инициализации, поэтому изменять его не требуется.

Теперь вставьте шаг IEC, которому можно назначить одно или несколько локальных действий.

Для создания шага IEC следует нажать на панели инструментов кнопку 'Type of created step' (Тип создаваемого шага).



Рисунок 33: Выбор/отмена выбора шага IEC на панели инструментов

**Выделите переход 'T2' / выберите пункт контекстного меню 'Create transition behind' (Создать переход до шага)**

Name (Имя)	ActionStep
Min. retention time (Мин. время удержания)	T#5s
<b>Кнопка [Close] («Закрыть»)</b>	

#### Шаг 4: создание действия и назначение действия шагу

Для создания действий необходимо перейти на вкладку 'Actions' (Действия) и выбрать пункт контекстного меню 'Add Action' (Добавить действие) в столбце 'Name' (Имя) (раздел 'Declaration' (Объявление)).

В нашем случае действию присваивается имя 'SetToggle', и оно выполняет следующий программный код.

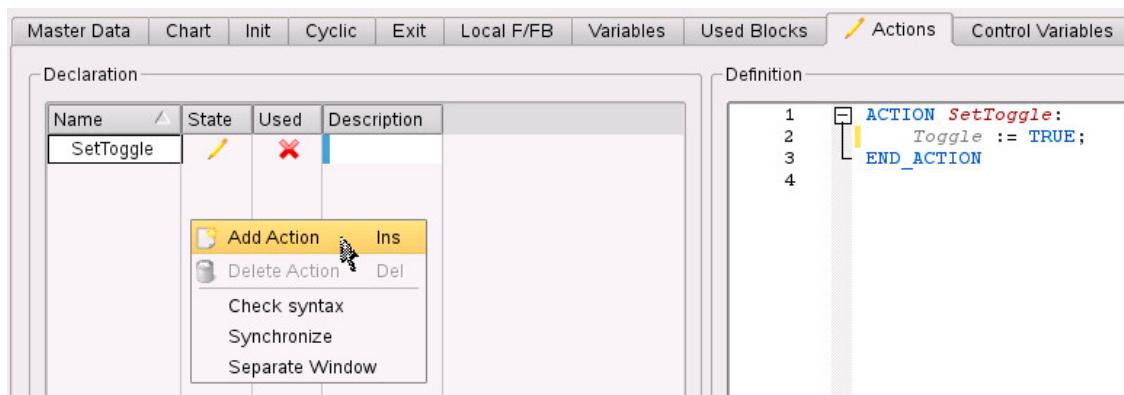


Рисунок 34: Настройка действия

Теперь можно назначить созданное действие шагу 'ActionStep'. Для этого нужно перейти к виду схемы, отметить шаг и открыть его свойства.

Также необходимо открыть редактор локальных действий в отдельном (separate) окне, используя панель инструментов.



Рисунок 35: Кнопка панели инструментов 'Local actions' (Локальные действия)

Теперь необходимое действие может быть назначено шагу путем простого перетаскивания из открывшегося списка действий на вкладку 'Called actions' (Вызываемые действия).

# SFC в разработке проектов

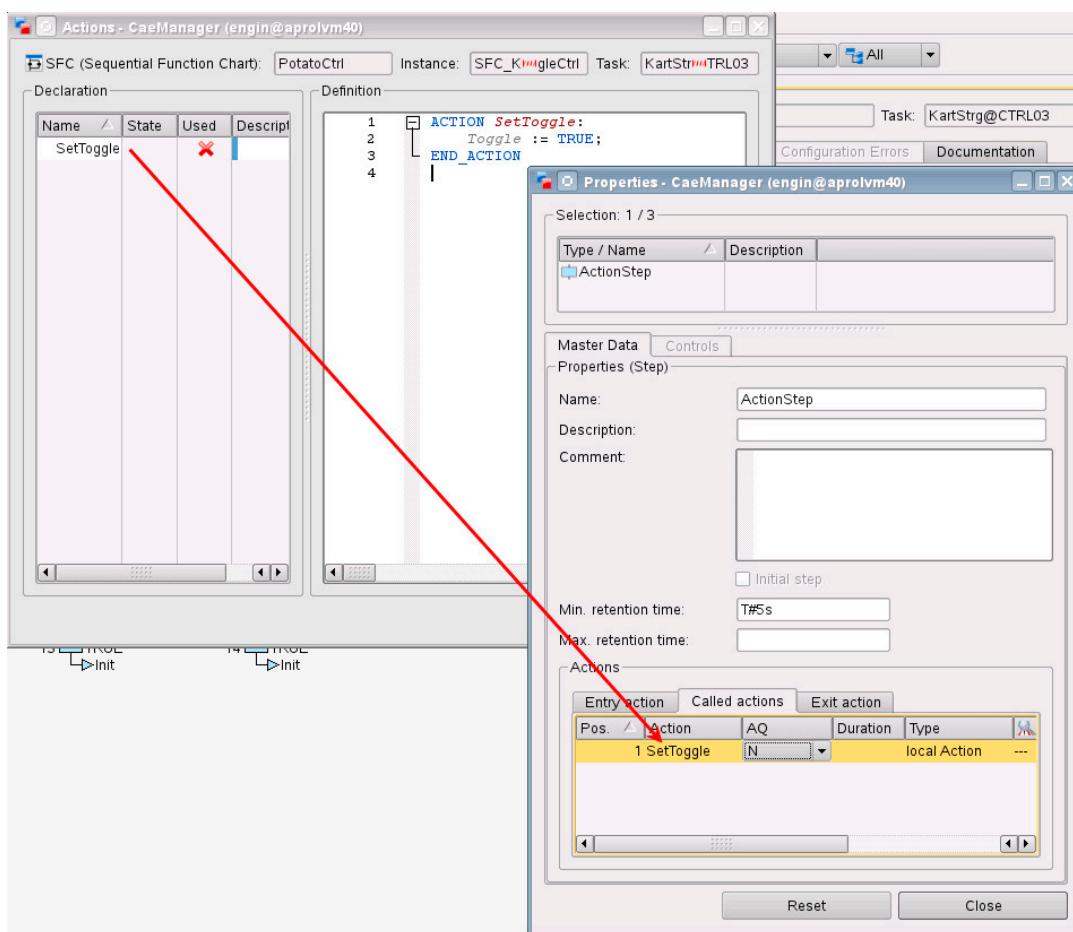


Рисунок 36: Назначение действия шагу IEC путем перетаскивания



Управление действиями в шагах также можно производить при помощи определителей действий, см. [Действия в шагах IEC](#)

**Теперь завершим создание программы SFC, выбрав пункт меню 'File/Save' (Файл/Сохранить), и выполним компиляцию, нажав [F8].**

## 5.3.3 Мониторинг времени работы шагов

Время работы каждого шага можно при необходимости отследить. Это позволяет проверить, не были ли нарушены допустимые верхний или нижний пределы. Нарушение заданных временных лимитов можно отследить с помощью управляющих переменных (см. раздел [Использование управляющих переменных SFC](#))

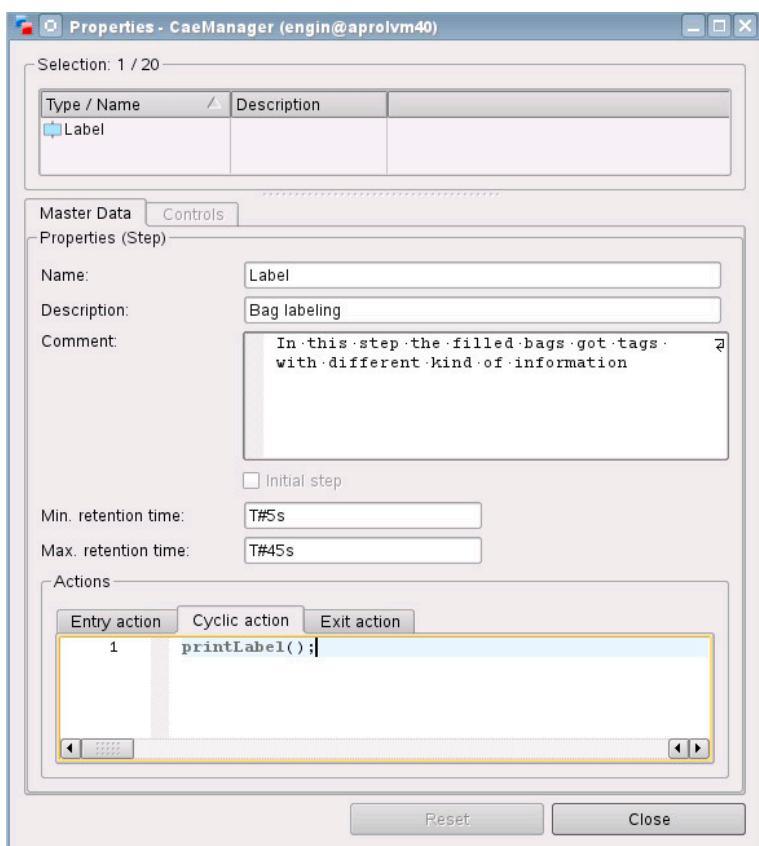


Рисунок 37: Диалог свойств шага с минимальным и максимальным временем пребывания.

**Минимальное время** – это минимальное время, в течение которого шаг должен являться активным.

**Максимальное время** – это максимальное время, в течение которого шаг может являться активным.



Для отслеживания нарушения временных лимитов выполнения шага из вне также необходимо указать управляющую переменную SFC '**SFCEnableLimit**'.

#### 5.3.4 Использование переменных в SFC

Можно объявить локальные переменные или выбрать для использования переменные проекта на вкладке '**Variables**' (Переменные).

Переменные проекта позволяют осуществлять обмен данными с другими программами (CFC, ST...)

##### **Переменные проекта:**

переместить переменную проекта можно путем простого перетаскивания (или двойного щелчка) из списка переменных в программу.

# SFC в разработке проектов

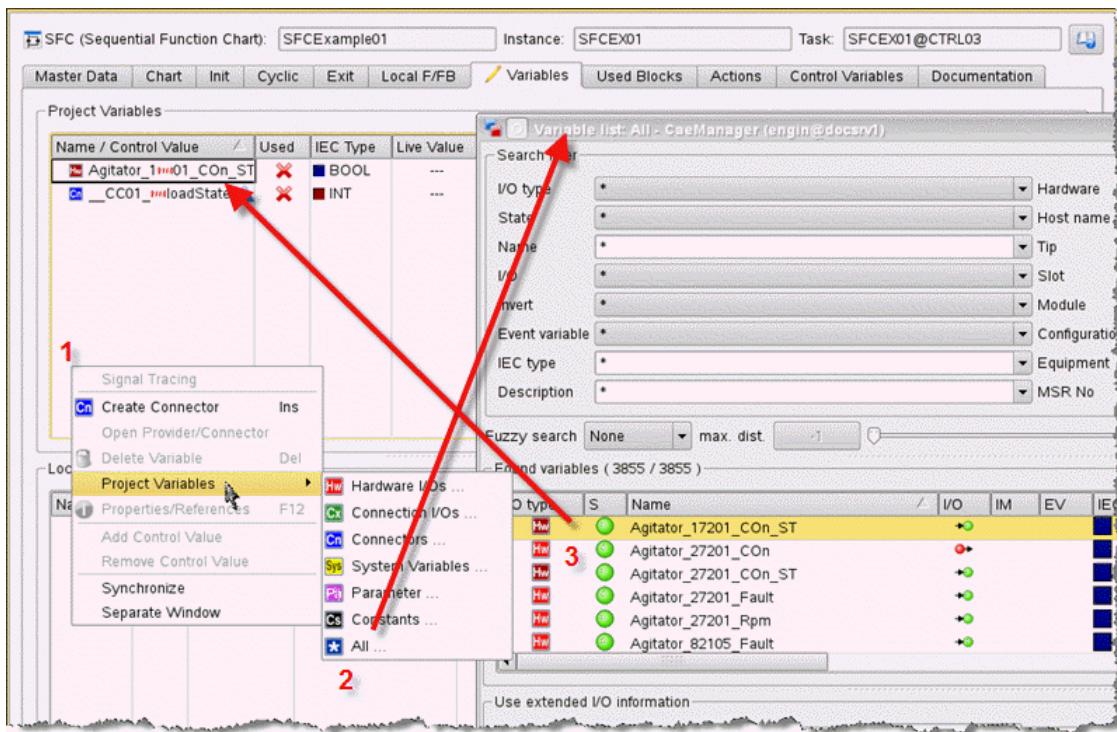


Рисунок 38: Вкладка 'Variables' (Переменные)

Помимо этого, для повышения эффективности процесса разработки проектные переменные можно использовать сразу в коде.

В процессе синхронизации вручную или автоматической синхронизации в ходе сохранения выполняется проверка, которая позволяет узнать, какие переменных уже используются, но ещё не были настроены. В этом случае проектные переменные будут добавлены в список проектных переменных на вкладке '**Variables**' (Переменные).

## Локальные переменные:

локальные переменные могут быть указаны как на вкладке '**Variables**' (Переменные) до того, как использоваться в коде ST, или сначала использоваться непосредственно в программном коде, а затем переноситься на вкладку '**Variables**' (Переменные) (в процессе синхронизации).

Обнаружение переменных, используемых в программном коде, но при этом еще не определенных, запускается при помощи пункта контекстного меню '**Synchronize**' (Синхронизировать) на вкладке '**Variables**' (Переменные).

По умолчанию создаются переменные типа 'BOOL'.

Для того чтобы сделать возможным создание задачи, необходимо явно определить локальные переменные, используемые при программировании.



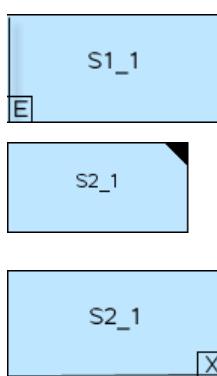
Способ использования и объявления локальных переменных в ST и SFC совпадает.

### 5.3.5 Действия в простых шагах

**Простые шаги** имеют одно входное действие, одно циклическое и одно выходное. Они хранятся в виде кода в диалоговом окне свойств шага.

Код ST может быть вставлен непосредственно на вкладки 'Entry action' (Входное действие), 'Cyclic action' (Циклическое действие) и 'Exit action' (Выходное действие).

Тип действия (входное, циклическое, выходное) также явно отмечается на схеме SFC:



#### Входное действие (E)

Действие, выполняемое **единожды при активации соответствующего шага**. Может быть описано на соответствующей вкладке.

#### Циклическое действие

Действие, которое выполняется **на протяжении периода активности соответствующего шага**. Может быть описано на соответствующей вкладке.

#### Выходное действие (X)

Действие, которое выполняется **единожды, когда шаг становится неактивным**. Может быть описано на соответствующей вкладке.



Выходное действие шага и входное действие **следующего шага** обрабатываются в пределах одного цикла.

### 5.3.6 Действия в шагах IEC

Настройка и отображение локальных действий, которые используются в пределах этой части проекта и могут быть назначены шагам IEC в сети SFC, осуществляются на схеме SFC в отдельном редакторе 'Local actions' (Локальные действия).

Настройку также можно выполнить на вкладке '**Actions**' (Действия).

Редактор делится на две области: область объявления и область определения. В левой части представлен обзор всех настроенных локальных действий (объявление).

В правой части (определение) находится текстовый редактор, отображающий код выбранного из списка действия . Здесь код может быть изменен и проверен на синтаксические ошибки.

#### Объявление:

Столбец	Краткое описание
Name (Имя)	Здесь вводится имя локального действия, которое используется в SFC для запуска действия в шаге.
Status of the action (Состояние действия)	Возможные состояния: - код действия был изменен - в ходе проверки синтаксиса в коде обнаружена ошибка - в ходе проверки синтаксиса было отображено предупреждение - в ходе проверки синтаксиса было отображено сообщение
Status of usage (Состояние использования)	Возможные состояния: - действие было назначено шагу SFC - действие не было назначено шагу SFC

# SFC в разработке проектов

Столбец	Краткое описание
Description (Описание)	Здесь пользователь может ввести описание действия.

Операции с действиями доступны при помощи контекстного меню, например вставка или удаление локальных действий:

Пункт меню	Описание
Add Action (Добавить Действие)	Добавление нового действия. Имя действия указывается автоматически.
Remove Action (Удалить действие)	Удаление выбранного действия из списка локальных действий. При удалении действия из списка локальных действий с помощью этого пункта контекстного меню появится диалоговое окно, в котором пользователь может выбрать удалять ли существующие назначения.
Check syntax (Проверить синтаксис)	Выполнение проверки синтаксиса кода <b>всех действий</b> .
Synchronize (Синхронизировать)	Выполняется синхронизация используемых локальных действий.

## Определение:

для определения алгоритма выбранного действия используется текстовый редактор. Редактор содержит все обычные функции APROL для комфортной работы.

## Для действий используется следующий синтаксис:

```
ACTION <Aktionsname>
  ...
END_ACTION
```

Синтаксис автоматически определяется системой, поэтому текущий код ST необходимо добавлять внутри автоматически созданного шаблона.

Диалоговое окно 'Action selection' (Выбор действия) открывается при помощи соответствующей кнопки на панели инструментов. В диалоговом окне можно выбрать булевы и локальные действия, доступные этой в части проекта.



Рисунок 39: Открытие диалогового окна назначения действий на панели инструментов

Действия, доступные для выбора и отображающиеся в диалоговом окне, отсортированы по типу и могут быть назначены шагам IEC путем перетаскивания.

Шаги, у которых есть назначенные действия, перечислены в нижней части окна.

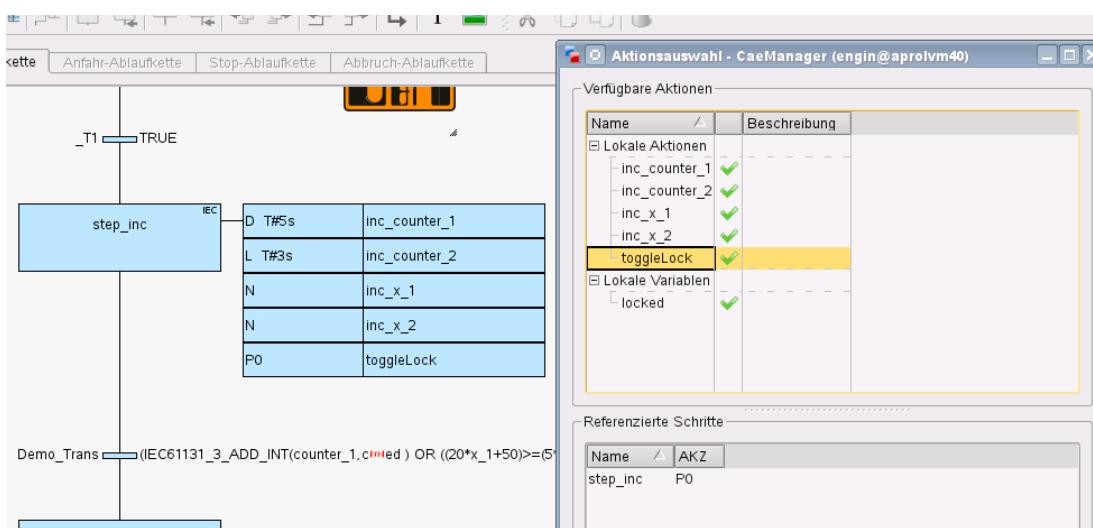


Рисунок 40: Выбор булевых или локальных действий

### Как использовать булево действие в шаге IEC?

Шаг	Описание
1	Откройте список проектных и локальных переменных с помощью соответствующей кнопки на панели инструментов вкладки 'Chart' (Схема).
2	Откройте диалоговое окно свойств шага IEC, в который необходимо вставить булево действие. Выберите вкладку 'Called actions' (Вызываемые действия).
3	Перетащите необходимую переменную (только типа BOOL) на вкладку 'Called actions'.



**Аппаратные вводы/выводы** могут использоваться в шаге IEC только как булевые действия через локальные переменные. После настройки сопоставления необходимо назначить локальную переменную как булево действие, а не переменную проекта.

### Что такое определители действия?

Определители действия (Action qualifiers, AQ) определяют условие активации для соответствующего действия, т. е. активация или деактивация действия, или значение, которое присваивается булевой переменной, всё это контролируется через AQ. Условие деактивации применимо только в случае, если действие уже активно.

При помощи определителей действия могут сохраняться, не сохраняться, запускаться по сигналу или запускаться по прошествии определенного времени.



Полный обзор всех определителей действий можно найти в документации по APROL, раздел **B2 Project Engineering / Logic creation with SFC / How are steps and actions used in an SFC? / Actions in IEC steps** (Разработка проекта B2 / Создание логики с использованием SFC / Использование шагов и действий в SFC) / Действия в шагах IEC

## SFC в разработке проектов

### 5.3.7 Использование управляющих переменных SFC

Для любой программы SFC при помощи объявления и использования управляющих переменных SFC можно активировать дополнительные функции (например, мониторинг времени для шагов).

Внутренние управляющие переменные SFC могут быть сопоставлены с проектными переменными процесса для использования их в логике, например, чтобы запись таких переменных осуществлялась при помощи внешней логики.

Создание переменных проекта выполняется при помощи вкладки 'Control Variables' (Управляющие переменные) функции '**Create Project Variables Automatically**' (Создавать проектные переменные автоматически).

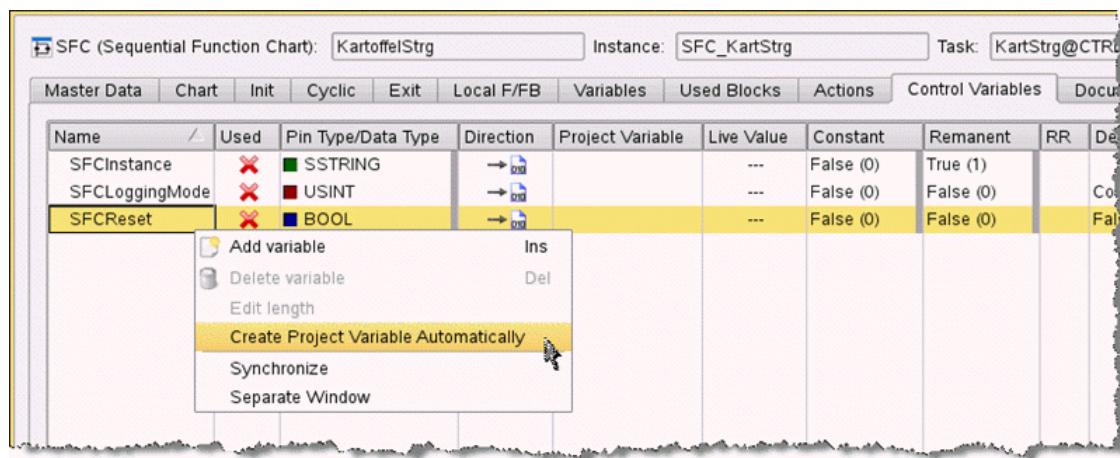


Рисунок 41: Пункт контекстного меню 'Create Project Variables Automatically' (Создавать проектные переменные автоматически)

В появившемся диалоговом окне можно ввести префикс для имен проектных переменных. В поле ввода по умолчанию автоматически подставляется имя экземпляра SFC.

Имя проектной переменной формируется следующим образом:

<префикс>\_<имя управляющей переменной>

Если проектные переменные уже были определены с другими именами, пользователю будет предложено подтвердить перезапись существующих переменных в отдельном диалоговом окне.

**Запись всех переменных типа 'input' может осуществляться логикой.**



Если запись управляющих переменных осуществляется логикой, конфигурация прав оператора для программы SFCViewer не имеет силы.

Таким образом, права оператора, используемые в проекте, необходимо настраивать в визуализации.



Таблица, содержащая все доступные действия, приведена в документации по APROL, раздел **B2 Project Engineering / Logic creation with SFC / How are control variables used in an SFC? (Разработка проекта B2 / Создание логики с использованием SFC / Использование управляющих переменных в SFC)**.

### 5.3.8 Упражнение – дисперсионный смеситель

Необходимо выполнить настройку смесителя, смещающего воду и колер.

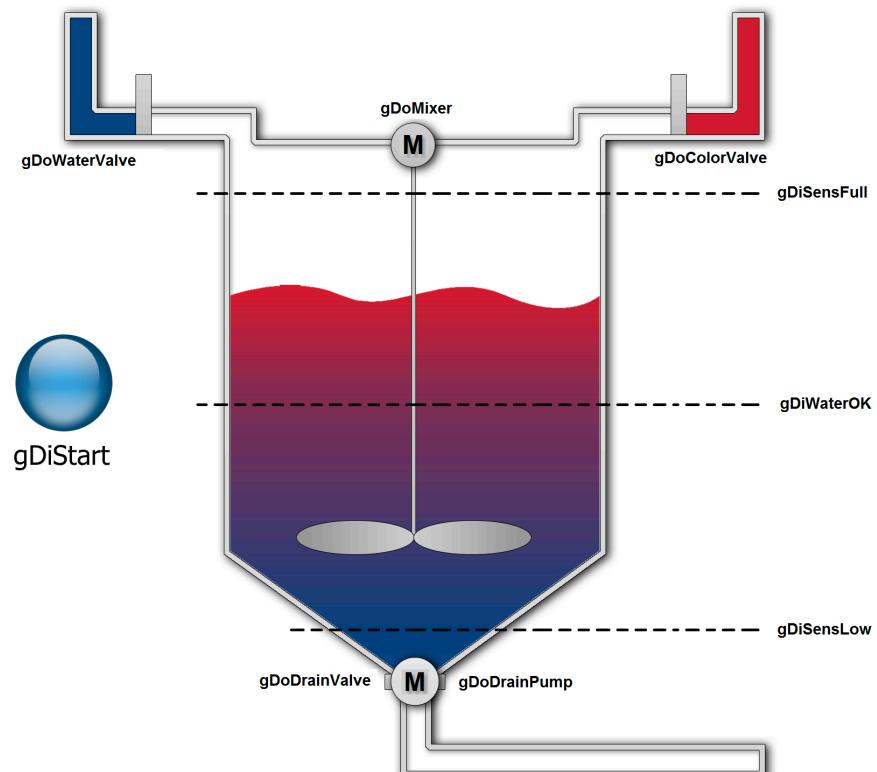


Рисунок 42: Схема системы смешивания краски

**Программа смещивания будет выполняться в соответствии со следующей процедурой:**

- Программа смещивания ожидает нажатия кнопки запуска (**gDiStart**).
- Вода (**gDoWaterValve**) добавляется в контейнер до тех пор, пока не сработает датчик "**gDiWaterOK**".
- Происходит запуск мешалки (**gDoMixer**) и добавление краски (**gDoColorValve**) до тех пор, пока не сработает датчик "**gDiSensFull**".
- Смешивание должно занять 30 секунд.
- После этого для слива готовой краски из емкости включается сливной клапан (**gDoDrainValve**) и сливной насос (**gDoDrainPump**).
- Процесс слива завершается при срабатывании датчика "**gDiSensLow**".
- Цикл приходит к своему началу.

## **Задача: реализация управления смесителем**

Теперь необходимо создать программу для описанной выше процедуры. Необходимо выполнить следующие пункты:

- 1) Создать эскиз необходимой SFC.
  - Определить шаги.
  - Определить переходы.
  - Определить действия.
- 2) Определить, какие шаги требуют наличия действия IEC.
- 3) Реализация требований в программе SFC



Мешалка должна быть активна в течение нескольких шагов.

**Эта функция может быть реализована различными способами:**

- Включением и выключением мешалки во входном и выходном действиях шагов
- Реализацией работы мешалки в качестве шага, параллельного другим шагам
- Включением мешалки при помощи определителя действия "S" и ее отключения в конце последовательности действием, имеющим определитель "R".

Пример реализации можно найти в приложении, в разделе [Пример программ SFC для решения задач из текста модуля](#).

## 5.4 Диагностические функции в SFC

### 5.4.1 Режим монитора в редакторе SFC

Обязательными условиями для отладки SFC на контроллере являются существование базы данных отладки в среде поддержки исполнения и наличие доступа к базе данных разработки.



Рисунок 43: Активация отладки

После выбора режима отладки (на панели инструментов или с помощью комбинации клавиш **[Ctrl] + [D]**), текущее состояние SFC на контроллере будет отображаться посредством выделения шагов, действий и переходов в редакторе SFC цветом.

При активации режима отладки на всех соответствующих вкладках будут отображаться значения переменных для отладки.

Если необходимые переменные состояния недействительны, соответствующие шаги и действия помечаются красным крестиком, а значения для отладки не отображаются.

Для того чтобы разрешить программе отладки SFC **управлять последовательностью действий**, на вкладке "Основные данные" SFC необходимо включить параметр 'Controllable by SFCViewer' (Управляется SFCViewer).

Активация параметра позволяет выбирать средства управления в SFCViewer и редакторе SFC.

Режим отладки может быть активирован (при помощи панели инструментов или **[Ctrl] + [D]**) после того, как часть проекта 'SFC' была активирована, скомпилирована, была произведена сборка и загрузка соответствующей задачи на контроллер.

Если **SFCViewer запускается в среде поддержки исполнения**, необходимо настроить права оператора для отладки SFC при помощи OperatorManager.

Никаких дополнительных прав, кроме "Sequential Function Chart: View" (Последовательная функциональная схема: просмотр) не требуется **для отладки SFC в CaeManager** (с помощью SFC редактора).

(Меню 'Extras / User Management / User / tab 'Rights Management' / Project rights 'SFCViewer" (Дополнительно / Управление пользователями / Пользователь / Вкладка 'Управление правами' / Проектные права 'SFCViewer'))

Настроить цветовое выделение объектов и вида SFC в соответствии с предпочтениями пользователя можно в CaeManager в разделе ('**Extras / User Options / 'General (Дополнительно / Пользовательские настройки / вкладка «Общие»)**').

#### Параметры цвета для вида отладки:

цветовое отображение шагов, которые были/не были выполнены на данный момент, активных шагов и действий, объектов, значение которым присвоено принудительно, а также фона SFC можно указать **для всего проекта** на вкладке 'Runtime options (2)' (Параметры поддержки исполнения (2)).

# SFC в разработке проектов



Влияние на программу или состояние элементов SFC идентичен функциям/конфигурации, описанным для SFCViewer. Подробную информацию см. в разделе [SFCViewer](#).

## 5.4.2 SFCViewer

Диагностика SFC и оказание влияния на нее выполняются при помощи программы SFCViewer. Эта программа может быть открыта в среде разработки и среде исполнения.

Помимо прочего, может потребоваться открыть несколько SFC одновременно (т. е. в многоэкранной среде).

Настройка экземпляров (макс. 5) осуществляется в CaeManager, объекта 'APROL system', раздел 'System monitoring' (Мониторинг системы), пункт контекстного меню SFCViewer 'New Instance' (Новый экземпляр).

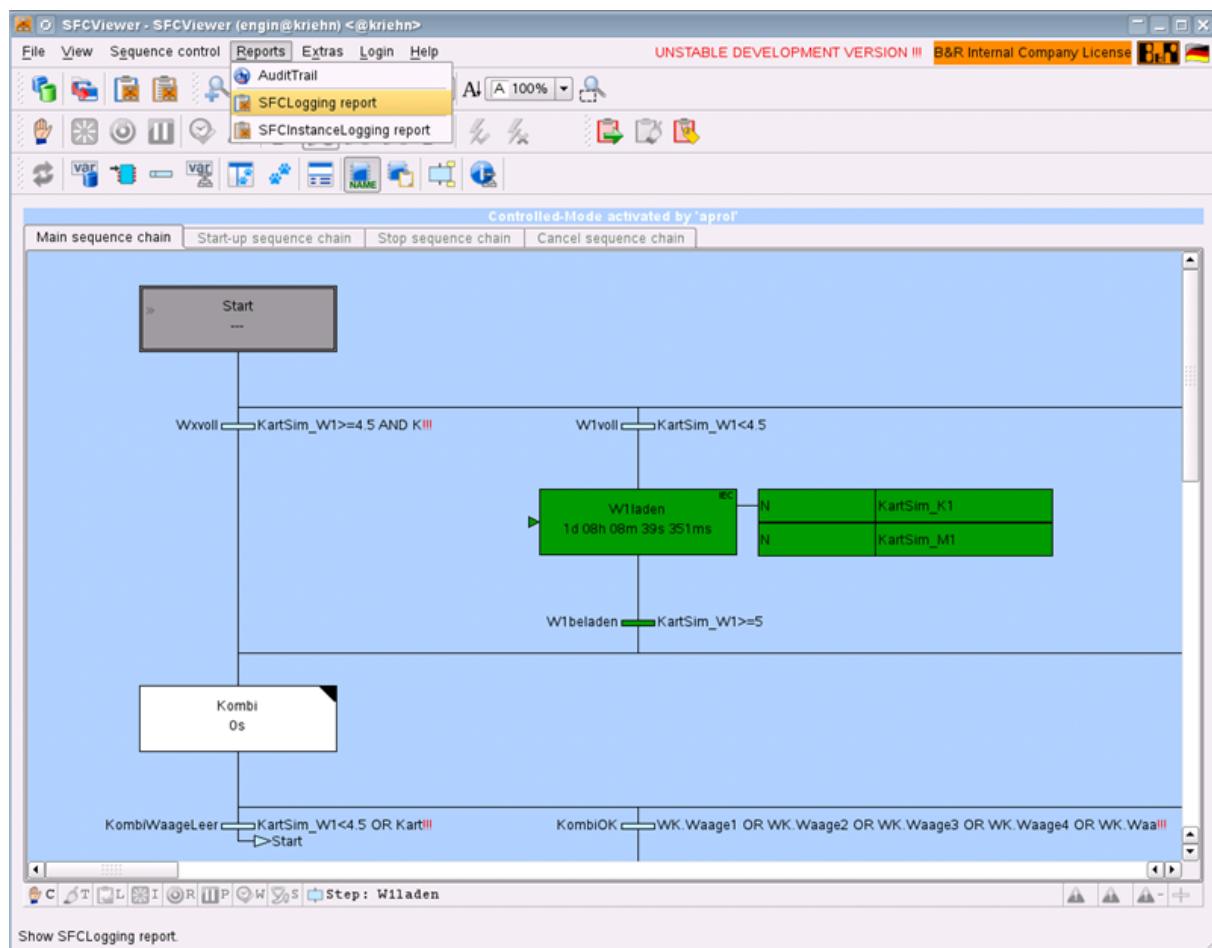


Рисунок 44: SFCViewer

## Онлайн-отладка функциональных блоков (SFC)

Наличие экземпляра блока в программе SFC или CFC, т. е. в проекте САЕ, является обязательным требованием для отладки функциональных блоков (SFC).

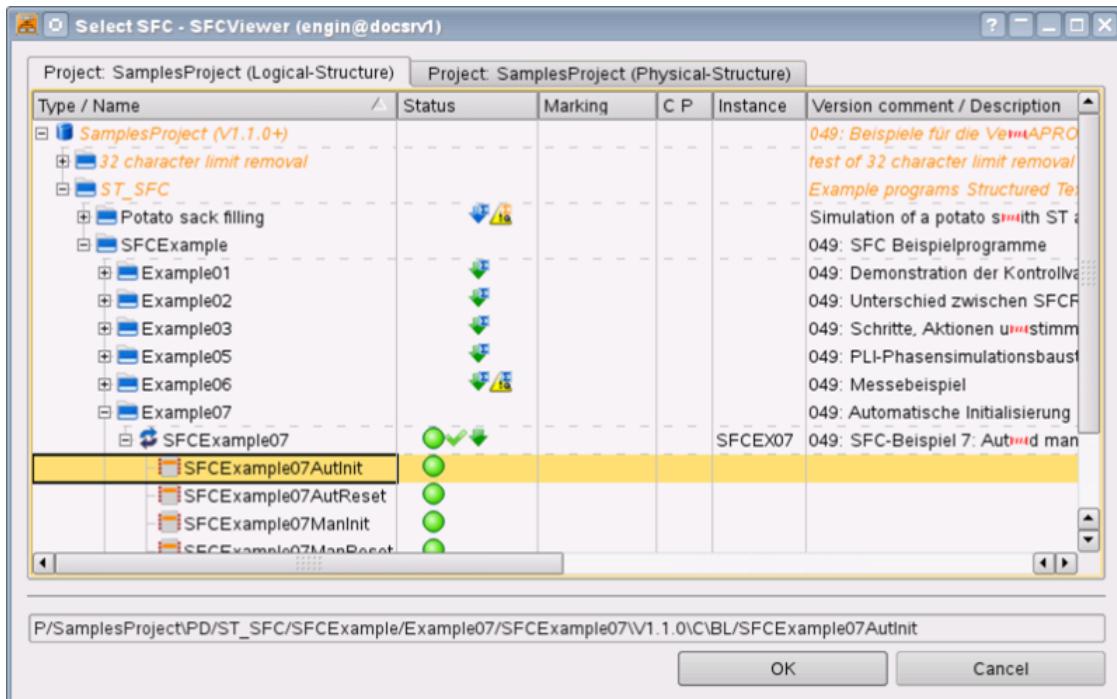


Рисунок 45: Выбор функционального блока (SFC) в SFCViewer

Откройте диалоговое окно выбора SFC (пункт меню 'File>Select SFC' (Файл/Выбрать SFC)) и выберите экземпляр функционального блока (SFC).

После активации режима отладки в SFCViewer можно начинать онлайн-отладку (в зависимости от предоставленных прав).

#### 5.4.2.1 Влияние на ход выполнение SFC

Осуществление влияния на последовательное управление происходит путем включения 'Controlled-Mode' (Управляемого режима).

Таким образом SFC переводится в состояние, в котором через активированные элементы управления (в зависимости от предоставленных прав доступа) осуществляется влияние на разработанную последовательность.



Иллюстрация: активация отладки и последовательного управления

#### Возможные действия для последовательного управления SFC:

Иконка	Описание возможных действий
	<b>Controlled Mode (Управляемый режим)</b> Включение управляемого режима для влияния на последовательное управление данной SFC.
	<b>Init (Инициализация)</b>

Иконка	Описание возможных действий
	Переводит SFC обратно к начальному шагу. Начальный шаг является активным до тех пор, пока активен параметр 'Init', но его обработка не происходит; мониторинг времени отключен. Нормальная обработка SFC возобновляется при возврате параметру 'Init' значения FALSE. Таким образом, оценка перехода при активном параметре 'Init' также не выполняется.
	<b>Pause (Пауза)</b> Останавливает обработку SFC. Состояние SFC остается неизмененным. Мониторинг времени отключен до тех пор, пока данный параметр является активным. При изменении состояния на 'SFC pause' (Пауза SFC) оценка переходов, сохраняемые и несохраняемые действия откладываются, т. е. останавливаются.
	<b>Time monitoring (Мониторинг времени)</b> Включение/выключение мониторинга времени для выполнения шага. Если параметр включен, превышения времени выполнения регистрируются в управляющей переменной 'SFCError'.



Это лишь выдержка из списка возможных действий для последовательного управления. Полный список содержится в документации по продукту APROL, раздел **B2 Project Engineering / Logic creation with SFC / How can the course of an SFC be monitored in the CaeManager? (Разработка проектов B2 / Создание логики при помощи SFC / Как выполняется мониторинг последовательности SFC в CaeManager?)**

Как в среде разработки, так и в среде исполнения доступны различные права для **влияния на ход выполнение SFC**.

#### 5.4.2.2 Влияние на состояние объекта SFC посредством принудительной активации или деактивации

Для выполнения различных процедур отладки отдельные шаги, переходы и действия могут «принудительно активироваться» или «принудительно деактивироваться» с учетом соответствующих прав доступа.



Возможность активации определенных шагов полезна, например, в ходе ввода в эксплуатацию. С ее помощью SFC можно перевести в определенное состояние.

Эта функция не сама включает форсирование (т. е. не влияет напрямую на шаги, переходы и действия), поскольку **действительное форсирование выполняется при помощи функции 'Confirm all force states' (Подтвердить все принудительные состояния)** в контекстном меню или на панели инструментов.



Обязательным условием для форсирования является включенный режим **'Controlled-Mode'**.

Форсирование может быть снято таким же образом – при помощи пунктов контекстного меню 'Remove force requests' (Удалить все запросы на форсирование) и 'Remove all force states' (Удалить все принудительные состояния).

### Особые функции при форсировании действий

В ходе форсирования действий необходимо различать **принудительное указание общего состояния действия** (принудительная деактивация/активация) и **принудительное указание состояния в контексте шага** (только принудительная деактивация).

**Принудительное указание общего состояния действия** присваивает одному действию, связанному с несколькими шагами, одно и то же состояние (активно/неактивно) во всех шагах.

**Принудительное указание состояния в контексте шага** ограничивает принудительную деактивацию действий выбранным шагом. Когда шаг находится в активном состоянии, данное действие не становится активным в контексте данного шага; другие экземпляры этого же действия в контексте других шагов, тем не менее, являются активными.

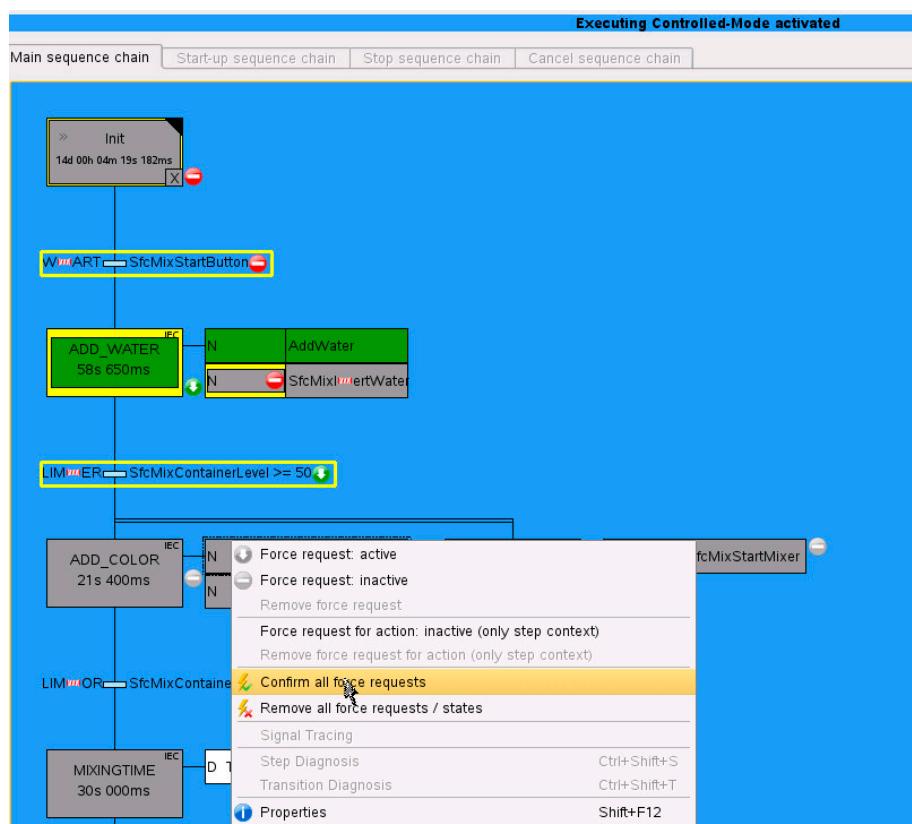


Рисунок 46: Принудительное указание состояния шагов, переходов и действий

Действие	Краткое описание
Force request: Active (Запрос на форсирование: в активно)	Принудительно осуществляет выполнение выбранного элемента, перезаписывая условие перехода значением TRUE. Требует обязательного подтверждения функцией 'Confirm all force requests' (Подтвердить все принудительные запросы).

Действие	Краткое описание
Force request: Inactive (Запрос на форсирование: в неактивно)	Принудительно осуществляет блокировку выбранного элемента, перезаписывая условие перехода значением FALSE. Требует обязательного подтверждения функцией 'Confirm all force requests' (Подтвердить все принудительные запросы).
Remove force request (Удалить запрос форсирования)	Удаляет запрос на форсирование для выбранного элемента.
Force request for action: inactive (only step context). Запрос принудительной деактивации действия (только в контексте шага)	Осуществляет блокировку выбранного действия во время активации соответствующего шага. Требует обязательного подтверждения функцией 'Confirm all force requests' (Подтвердить все принудительные запросы).
Remove force request for action (only step context). Удалить запрос на принудительную деактивацию действия (только в контексте шага)	Удаляет принудительные состояния для действий в контексте шага. Требует обязательного подтверждения функцией 'Confirm all force requests' (Подтвердить все принудительные запросы).
Confirm all force requests (Подтвердить все запросы на форсирование)	Подтверждает и активирует все принудительные состояния.
Remove all force state (Удалить все принудительные состояния)	Удаляет все принудительные состояния.

### 5.4.2.3 Влияние на состояние объекта SFC при помощи режима подсказок

Режим подсказок – это второй способ оказания влияния на последовательное управление SFC. Переключение в «режим подсказок» позволяет оператору осуществлять контролируемый пошаговый запуск последовательности в процессе обычного выполнения SFC.



Обязательным условием для режима подсказок также является работа в режиме 'Controlled-Mode'.

Включение режима подсказок осуществляется на панели инструментов. Задав описанные ниже параметры, станет возможно выполнить программу SFC в режиме подсказок.

Значок подсказки перед шагом показывает, что шаг перешел к выполнению с использованием «подсказки».

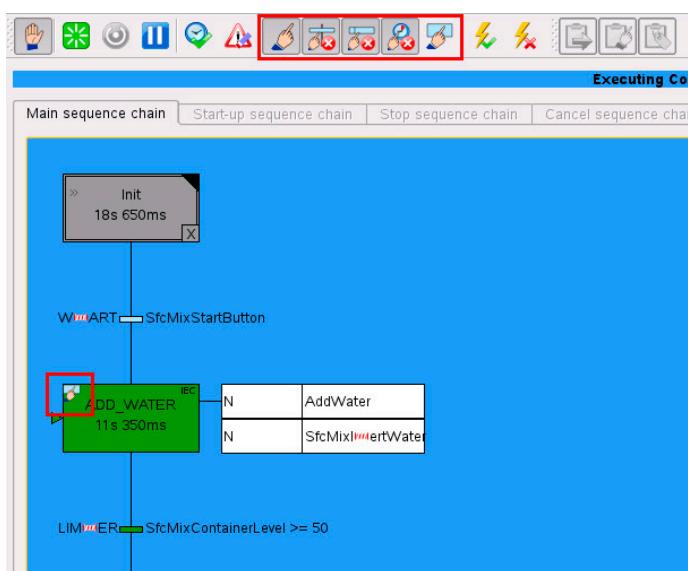


Рисунок 47: Работа в «режиме подсказок»

Действие	Краткое описание
Tip mode (Режим совета)	Активация/деактивация режима подсказок для данной SFC. Если режим активирован, все условия перехода считаются имеющими значения FALSE. При использовании «подсказки» переход от одного шага к следующему осуществляется вручную.
Ignore transition condition during 'tip' (Игнорировать условие перехода в режиме подсказок)	Включение/отключение учета условия перехода при работающем режиме подсказок. Примечание: Если параметр включен, при переходе от шага к шагу вручную при помощи «подсказки» условие перехода не будет приниматься во внимание.
Do not execute actions when tipping (Не выполнять действия в режиме подсказок)	Параметр включает/отключает выполнение действий в шаге, активированном «подсказкой», при включенном режиме подсказок.
Ignore dwell time during 'tip' (Игнорировать время выполнения в режиме подсказок)	Параметр включает/отключает контроль минимального времени выполнения шагов, активированных подсказкой, при включенном режиме подсказок.
Tip (Ручной режим)	Если параметр 'Ignore transition condition during tip' отключен, то переход к следующему шагу выполняется только при выполнении условия перехода. В противном случае переход выполняется без учета условия перехода.



При выполнении в режиме подсказок сети SFC, имеющей альтернативные ветви, переход (если все условия имеют значение TRUE или включен параметр для их игнорирования) выполняется всегда к крайней левой ветви ввиду того, что она имеет наивысший приоритет.

### **Задача: выполнение программы SFC в режиме форсирования и в режиме подсказок**

По завершению и загрузке предыдущего задания, посвященного управлению смесителем, откройте соответствующую программу SFC в SFCViewer.

Протестируйте различные функции, имеющиеся в SFCViewer. Например, можно **принудительно активировать/деактивировать** различные объекты или перемещаться по сети SFC в режиме **подсказок**.

#### 5.4.2.4 Блокировка управления SFC

SFC можно управлять посредством активации 'Controlled-Mode' (Управляемого режима). В этом режиме сигналы с логики игнорируются, и оператор таким образом осуществляет прямое влияние на ход выполнения последовательности и состояния объектов.

Два оператора, просматривающие одну SFC в управляемом режиме, оба имеют возможность вмешиваться в ее выполнение, и скорее всего их действия будут конфликтовать.

Во избежание таких ситуаций был создан механизм блокировки, предоставляющий оператору исключительный доступ к управляемому режиму и, следовательно, возможность влиять на SFC. Эксклюзивный доступ оператора к управляемому режиму называется «'executing Controlled-Mode руководящим управляемым режимом'».

Перехват управляемого режима SFC может осуществляться в любое время, **как с отображением запроса для оператора, работающего в данный момент, так и без**.



**Немедленный перехват управляемого режима** запрашивающим оператором **можно произвести в любой момент**, даже если текущий оператор не отреагировал на всплывающий запрос.

**Механизм блокировки обладает следующими преимуществами:**

- Безопасность в процессе совместного доступа**  
Исключительный доступ оператора к управляемому режиму («руководящий управляемый режим»)
- Возможность перехвата выполняемого управляемого режима**  
Перехват с отправкой запроса оператору, в данный момент управляющему процессом  
**Немедленный перехват управляемого режима** (гарантированный перехват при аварийных ситуациях)
- Обзор управляемого режима**  
Обзор всех управляемых SFC в одном диалоговом окне

### Выполнение в управляемом режиме

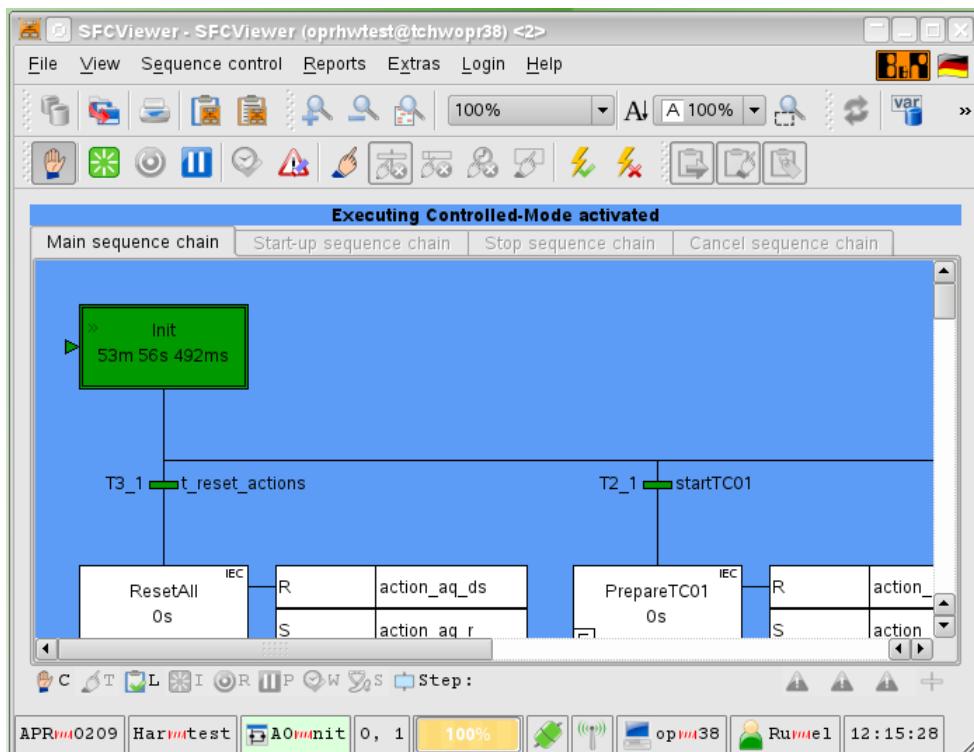
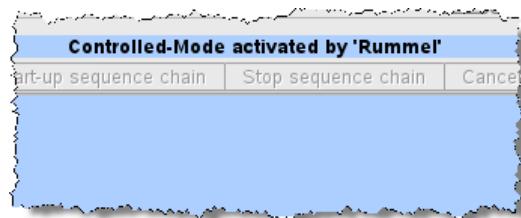


Рисунок 48: Отображение руководящего управляемого режима

Оператор, первым активировавший управляемый режим в SFC, первым получает к нему доступ.

Другой оператор получает информацию о том, какой оператор активировал управляемый режим, в виде сообщения о состоянии в верхней части представления схемы.



#### 5.4.3 Графическая диагностика переходов в SFC

Графическая диагностика переходов позволяет просмотреть входные значения для условия перехода; посредством анализа условий можно получить представление о состоянии перехода.

Узнать причины, по которым переход не активируется, можно при помощи диагностики перехода.

Поскольку интерпретация сложных логических выражений в условии перехода является достаточно сложной задачей, условие перехода отображается графически (в виде блок-схемы).

Диагностика перехода соответствует выражению перехода, созданному на языке IEC 'ST'.

Как для режима отладки в **CaeManager**, так и для режима отладки в **SFCViewer** верно следующее:

для выбранного перехода отображаются все входные переменные, все значения промежуточных результатов и конечный результат.

Также режим диагностики показывает, был ли переход включен принудительно.

## SFC в разработке проектов

Графическая диагностика переходов содержит следующие удобные функции:

- Отображение описания входных переменных и функций библиотеки APROL во всплывающей подсказке для объекта.
- Принудительная активация переходов непосредственно из диагностики перехода.
- Пометка входных значений и промежуточных результатов, доступ к которым не может быть получен, т. е. при отсутствии соединения с контроллером.
- Соответствующий переход отмечается синей рамкой на схеме SFC в диалоговом окне 'SFC Transition Diagnosis' (Диагностика перехода SFC) при наведении на него указателя мыши. Таким образом идентификация необходимого перехода обеспечивается даже при одновременном открытии нескольких диалоговых окон 'SFC Transition Diagnosis'.

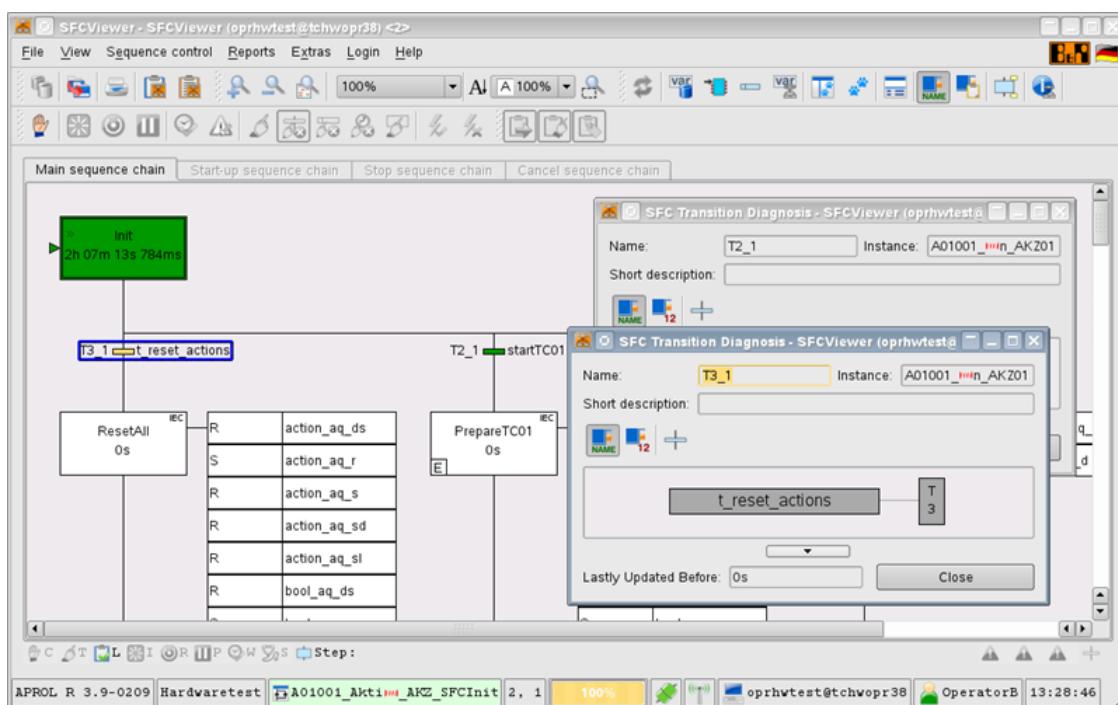


Рисунок 49: Диагностика перехода, оцениваемого в текущий момент

- В нижней части диалогового окна имеется поле отображения, в котором приведено время, прошедшее от последнего обновления отображаемого в данный момент перехода.
- Автоматическая настройка размера: при изменении размера диалогового окна 'SFC Transition Diagnosis' соответствующее графическое представление автоматически подготавливается по размеру к доступной области окна.

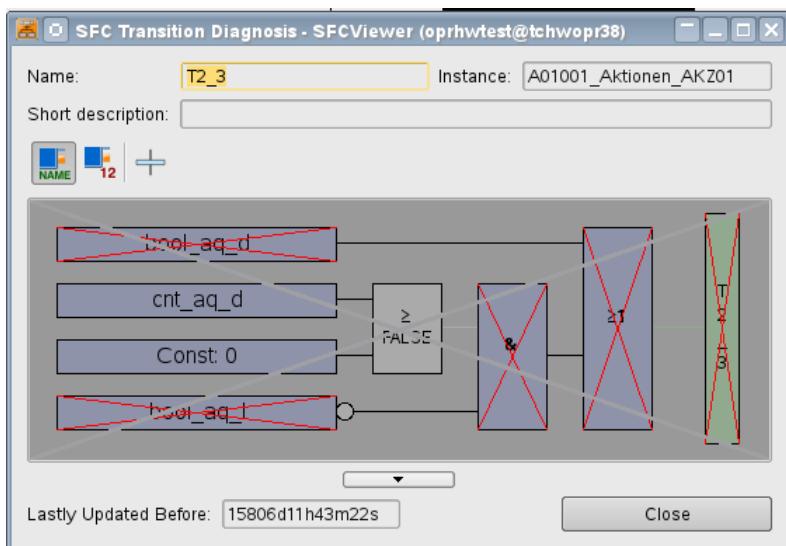


Рисунок 50: Графическая диагностика переходов в SFC

**Варианты открытия диагностики перехода:**

- выберите переходы, а затем выберите 'SFC Transition Diagnosis' (Диагностика перехода SFC) в контекстном меню - диагностика перехода отображается в виде блок-схемы.
- Щелкните правой кнопкой мыши в свободной области SFC и выберите 'SFC Transition Diagnosis' (Диагностика перехода SFC) в контекстном меню - диагностика перехода, обрабатываемого на данный момент, отображается в виде блок-схемы. При работе с альтернативными путями отображается крайний справа переход.
- SFCViewer:** двойной щелчок на переходе.
- CaeManager:** Двойным щелчком мыши на переходе **при включенном режиме отладки**.

**В виде кнопок на панели инструментов доступны следующие параметры отображения:**

Отображение имен входных переменных.

Отображение текущих значений входных переменных.

Переключение режима отображения между 'Display the last evaluated transition' (Отображать последний оцененный переход) и 'Display the last selected transition' (Отображать последний выбранный переход).

Графические элементы отображаются в сером цвете и кнопки панели инструментов недоступны, если режим отладки не является активным.

**В свойствах проекта (вкладка 'Runtime 2/2') можно настроить следующие цвета:**

- Цвет для булевых значений в состоянии 'FALSE'
- Цвет для булевых значений в состоянии 'TRUE'
- Цвет для числовых значений
- Цвет фона для перехода **в активном состоянии**
- Цвет фона для перехода **в неактивном состоянии**

# SFC в разработке проектов

## 5.4.4 Диагностика шагов

### Цели и преимущества

Все данные, относящиеся к текущему шагу SFC, отображаются в отдельном окне диагностики шага, в том числе имена переменных, описания, значения для отладки, ожидаемые значения и тип использования.

Таким образом, можно ответить на следующий вопрос: что происходит при выполнении шага?

#### Это означает следующее:

- результаты выполнения шага в контексте последовательной схемы или внешнем (при обмене данными с проектом) можно считать непосредственно с помощью диагностики шага SFC.
- Диагностика шага SFC отображает **значения до, в процессе и после выполнения шага**.

Диагностику шага SFC можно открыть в контексте шага, выбрав соответствующий пункт контекстного меню 'SFC Step Diagnosis' (Диагностика шага SFC). В режиме отладки или в SFCViewer диагностика открывается двойным щелчком по соответствующему блоку.

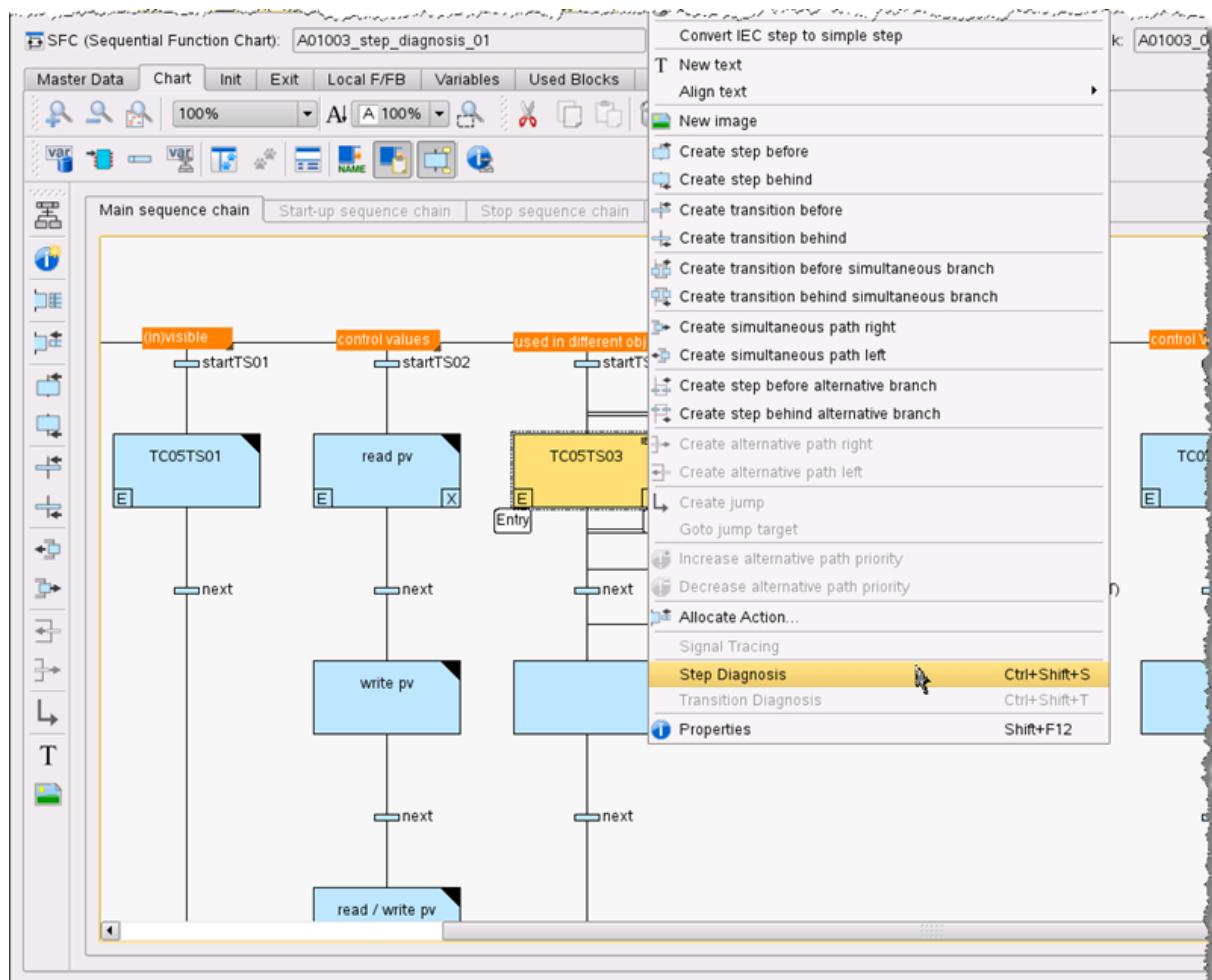


Рисунок 51: Открытие диагностики шага SFC



Диагностику можно открыть сразу для нескольких шагов (так же, как и диагностику для нескольких переходов), но только по одному экземпляру на шаг.

В случае если открыто нескольких диалоговых окон и одно из них активно, то соответствующий шаг выделяется на схеме голубым цветом.

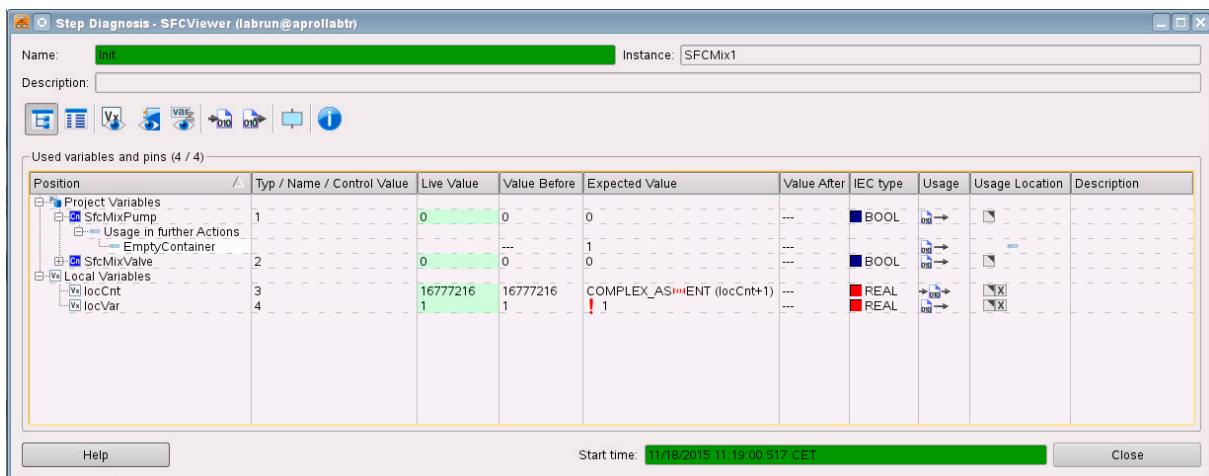


Рисунок 52: Открытая диагностика шага SFC

#### Отображение используемых переменных/контактов для шага

- Проектные переменные, используемые в программе, или сопоставленные с локальными переменными, отображаются в программах SFC.
- По умолчанию для функциональных блоков SFC отображаются только входные и выходные контакты в зависимости от их видимости.



**Настройка диагностики шага**, например, определение позиций, на которых отображаются различные переменные, а также общей видимости переменных в диагностике шага, выполняется при разработке/в редакторе SFC для соответствующего шага.

Если переменная сопоставлена с локальной, то в узле 'Project variables' (Переменные проекта) отображается только соответствующая проектная переменная. Имя локальной переменной выводится во всплывающей подсказке.

Для каждой переменной выделена отдельная запись, в которой отображается вся связанная с ней информация.

**Также для каждой переменной могут быть показаны следующие «дочерние элементы»:**

- Другие шаги, в которых используется переменная (включая направление чтения/записи)
- Другие действия IEC, в которых используется переменная (включая направление чтения/записи)
- Назначенные управляющие значения

## SFC в разработке проектов

Также на панели инструментов доступно множество возможностей фильтрации. Описание средств фильтрации можно найти в соответствующих всплывающих подсказках и документации по продукту APROL.

**Кнопки фильтрации можно использовать и в режиме отладки и в SFCViewer.**



Рисунок 53: Кнопки фильтрации на панели инструментов

**Помимо фильтров на панели инструментов содержатся следующие функции:**



**Отображение активного шага:**

активация 'Tracing mode' (Режима отслеживания), который отображает информацию о последнем активном шаге.

В одновременных цепях это шаг с «наиболее высоким приоритетом», т. е. шаг первого из параллельных путей.



**Свойства шага:**

отображается инспектор объектов для шага, открытого в средстве диагностики.

*Инспектор объектов для выбранного шага можно открыть так же при помощи контекстного меню.*

**Подробная информация о диагностике шага:**

детальное описание и объяснение отдельных столбцов можно найти в документации по APROL: B2 Project Engineering / Logic creation with SFC / Note about the implementation of an SFC (Разработка проекта B2 / Создание логики с использованием SFC / Памятка по реализации SFC).

Там же приведена информация о **табличной компоновке и выборе цвета** для сети SFC.

### 5.4.5 Функции регистрации действий в SFC

#### Каковы преимущества регистрации действий SFC в APROL?

APROL позволяет непрерывно регистрировать выполнение SFC.

#### Регистрация действий в SFC содержит следующие основные преимущества:

- Запись состояния шагов/действий/переходов
- Настройка SFC в CAE при помощи управляющих переменных
- Запись данных в отдельное хранилище ChronoLog
- Удобный анализ при помощи отчета 'SFCLogging'
  - Открытие отчета 'SFCLogging' в контексте проекта
  - Открытие отчета 'SFCLogging' в контексте SFC в SFCViewer
- Расширенные возможности фильтрации (по имени, экземпляру и т. д.)
- Создание пользовательских отчетов (с помощью языка запросов ChronoLog)



Для того чтобы можно было начать регистрацию данных в SFC, на контроллерах, выполняющих задачи SFC, должен присутствовать модуль 'ApCnfLog'. Информация о добавлении таких модулей на контроллер содержится в документации по продукту APROL:

**Manual 'B2 Project Engineering', chapter Logic creation with SFC (IEC 61131-3) / Which logging possibilities are supported for SFC? / How is the SFC logging configured in APROL? (Руководство 'Разработка проекта B2', глава 'Создание логики при помощи SFC (IEC 61131-3)' / Какие возможности регистрации поддерживаются для SFC? / Как выполняется настройка регистрации SFC в APROL?)**

Данные о последовательности, данные управления и контекстные данные SFC могут быть записаны выборочно. В процессе создания схем и блоков SFC для включения режима регистрации предлагается управляющая переменная '**SFCLoggingMode**', включающая регистрацию данных SFC или блока SFC в виде **двоичного кода**.



Подробная информация о данных последовательностей, управляющих данных и контекстных данных содержится в документации по APROL: **B2 Project Engineering / Logic creation with SFC / Which logging possibilities are supported for SFC?** (Разработка проекта B2 / Создание логики при помощи SFC / Какие возможности регистрации поддерживаются для SFC?)

### Отчет 'SFCLogging' в контексте SFC

Отчет SFC Logging можно открыть из меню KDE или из SFCViewer (на панели инструментов) в контексте SFC, мониторинг которой осуществляется на данный момент.

Time		D	E	S	C	Object name	Object type	Control	Value	Duration	Project	Instance
		↪	↪	↪	↪		Object				Project	
04.11.2015 06:28:25.399	01.01.1900 00:00:00	↪	↪	↪	↪	—	global	SFC instance	Set internal logging mode	Logging deactivated	---	AprolECamp SFC1
03.11.2015 15:50:49.915	18.11.2015 14:17:03.080421	↪	↪	↪	↪	—	global	SFC instance	Set controlled mode	Inactive	---	AprolECamp SFC1
03.11.2015 15:36:30.508	[ms] 100 Results / Page Project ...	→	○	○	○	—	_S1	Step	Inactive	---	Begin: 2015-11-03 15:36:25.458 Pause: 0m 0s Duration: 0m 5s	AprolECamp SFC1
03.11.2015 15:36:30.508		→	✳	✳	✳	—	Init	Step	Init step	---	Duration: 0m 14.250s	AprolECamp SFC1
03.11.2015 15:36:30.508		→	!	!	!	—	Init	Step	Active	Tip: ✖	---	AprolECamp SFC1
03.11.2015 15:36:30.458		→	✓	✓	✓	—	_T7	Transition	Transition proceeded	---	---	AprolECamp SFC1
03.11.2015 15:36:28.108		↪	⚡	—	✋	global	SFC instance	All force requests / states removed	---	---	---	AprolECamp SFC1
03.11.2015 15:36:25.458		→	!	!	!	—	_S1	Step	Active	Tip: ✖	---	AprolECamp SFC1
03.11.2015 15:36:25.458		→	○	○	○	—	_S5	Step	Inactive	---	Begin: 2015-11-03 15:36:22.408 Pause: 0m 0s Duration: 0m 3s	AprolECamp SFC1
03.11.2015 15:36:25.458		→	○	○	○	—	_S4	Step	Inactive	---	Begin: 2015-11-03 15:36:22.408 Pause: 0m 0s Duration: 0m 3s	AprolECamp SFC1

Рисунок 54: Отчет SFC Logging

#### 5.4.6 Отчет AuditTrail для SFC

Отчет AuditTrail можно, как и прочие отчеты, открыть с помощью элемента меню '**Reports (Отчеты) / AuditTrail**'. Удобная фильтрация осуществляется при помощи группы действий 'SFC intervention' (Корректировки SFC) и выбранных операций переключения.

## SFC в разработке проектов

The screenshot shows a Mozilla Firefox browser window titled "AuditTrail report - Mozilla Firefox". The address bar displays the URL: <https://tchwopr38.global.br-automation.com/PROJECTS/Hardwaretest/001/standard/action/overview.ch?lang=001&id=USERACTION%3A&restricts=projectHardwaretest&action=login&action=logout&action=alarm>. The main content area is titled "AuditTrail report - Filtered" and shows a table of audit logs. The table has columns: Time, Action, Operator, Surname, Firstname, CC-Account, Server, Operator terminal, and Project. The data in the table is as follows:

Time	Action	Operator	Surname	Firstname	CC-Account	Server	Operator terminal	Project
04/11/2013 13:10:04	Controlled mode	OperatorB			oprhwtest	tchwopr38	tchwopr38:2.0	Hardwaretest
04/11/2013 13:10:03	Controlled mode transferred to another operator	OperatorB			oprhwtest	tchwopr38	tchwopr38:2.0	Hardwaretest
04/11/2013 13:10:00	Operator demands a transfer of the controlled mode	OperatorB			oprhwtest	tchwopr38	tchwopr38:2.0	Hardwaretest

Рисунок 55: Отчет по корректировкам SFC AuditTrail

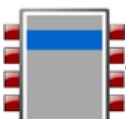
## 6 ST в разработке библиотек

### 6.1 Введение и функции редактора

В библиотеках APROL в целях образования единой структуры и для обеспечения повторного использования кода ST в пределах программы могут быть созданы следующие объекты:



Функции



Функциональные блоки

Функции и функциональные блоки имеют отрисовку в виде блока, хотя это не является обязательным требованием для использования в программе ST. Благодаря этому функции и блоки ST также могут использоваться в CFC при помощи графического программирования.



Помимо указания основных данных и вида блока редактор ST обеспечивает возможность настройки таких же параметров, как для программ ST в проекте. Соответствующая информация приведена в разделе 4.1 "Введение и функции редактора".

**Дальше будут описаны только вкладки, отличающиеся от вкладок для программы ST.**

### 6.2 Создание функционального блока ST

В контексте создания функций и функциональных блоков необходимо выполнение следующих действий:



Ниже приведен пример создания функционального блока (ST). Также можно создать функцию, написанную на ST.

Шаг	Разработка
1	Создание функционального блока ST
2	Настройка вкладки основных данных
3	Написание кода ST
4	Объявление вводов и выводов

## ST в разработке библиотек

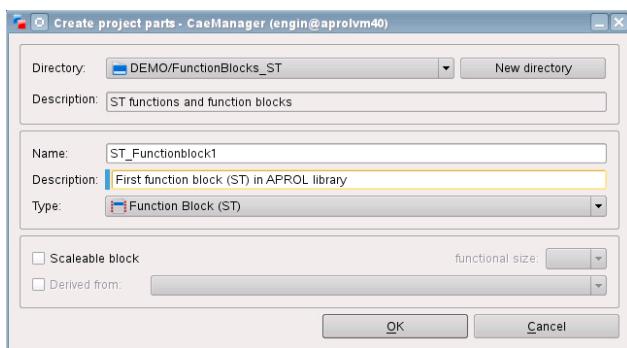


Рисунок 56: Создание функционального блока ST

**Навигация: CaeManager / CAE library / File / New / Function Block (ST) (CaeManager / Библиотека САЕ / Файл / Создать / Функциональный блок (ST))**



Name (Имя)

<имя блока>

Подтвердите нажатием кнопки [OK]

### 6.3 Основные данные функционального блока

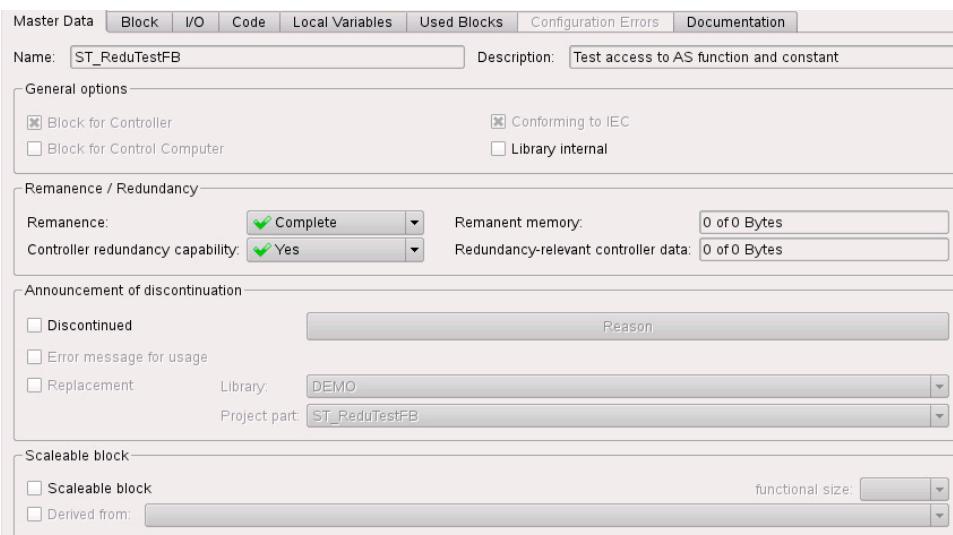


Рисунок 57: Вкладка "Основные данные" (Master Data)

На первой вкладке редактора ST "Master Data" (Основные данные) доступны следующие параметры, действующие в пределах библиотеки.

- Часть проекта ST всегда требует наличия контроллера и всегда соответствует IEC, поэтому соответствующие параметры всегда включены. Пользователь может только отметить эту часть проекта как внутреннюю для библиотеки.
- Remanence (энергонезависимость): здесь можно отметить весь блок, т. е. все его выводы, в качестве реманентного.
- Чтобы блок мог использоваться на резервируемом контроллере, необходимо включить параметр "Controller Redundancy Capability" (Совместимость с резервируемым контроллером).
- Ниже блок можно отменить, используя при этом сообщение об ошибке и указанием на замещающий блока; ещё ниже настаивается возможность масштабирования.

# ST в разработке библиотек

## 6.4 Использование контактов и переменных

Существует множество способов использования переменных в функциональном блоке (ST):

- В качестве вводов и выводов в представлении блока
- В качестве вводов и выводов в представлении списка вводов/выводов
- В качестве локальных переменных

### Вводы и выводы в представлении блока

Линии ввода и вывода (контакты) указываются так же, как всегда в APROL, на вкладке 'Block' (Блок), и таким образом создается компоновка блока.

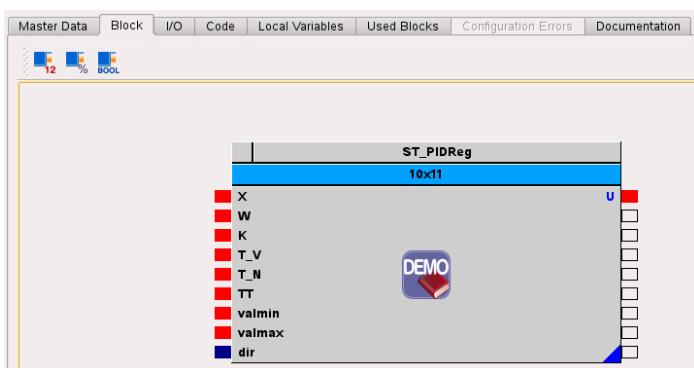


Рисунок 58: Представление блока для функционального блока (ST)

Для создания нового контакта выберите пункт контекстного меню "**Create New Pin (Создать новый пин)**" в свободной части блока. Если свободное место отсутствует, перетащите вниз нижнюю границу блока.

### Вводы и выводы в представлении списка вводов/выводов

Все входные и выходные контакты блока перечислены на вкладке 'I/O (Ввод/вывод)'. Новые контакты можно создать при помощи пункта контекстного меню '**Insert new pin**' (Вставить новый контакт).

Pos.	Name	IEC Type / Pin Type	Default Value	Unit	Description	Trans.	RR	Dyn.	Sys.
1	X	REAL	0		049:	✗	✗	✓	✗
2	W	REAL	0		049:	✗	✗	✓	✗
3	K	REAL	0		049:	✗	✗	✓	✗
4	T_V	REAL	0		049:	✗	✗	✓	✗
5	T_N	REAL	0		049:	✗	✗	✓	✗
6	TT	REAL	0		049:	✗	✗	✓	✗
7	valmin	REAL	0		049:	✗	✗	✓	✗
8	valmax	REAL	0		049:	✗	✗	✓	✗
9	dir	BOOL	False (0)			✗	✗	✓	✗

Pos.	Name	IEC Type / Pin Type	Unit	Description	Rem.	RR	Dyn.	Sys.	Vis.	Co.
1	U	REAL			✓	✗	✓	✗	✓	

Рисунок 59: Представление ввода/вывода для функционального блока (ST)

Функции на этой вкладке аналогичны функциям предыдущей вкладки 'Block' (Блок).

### Локальные переменные

Переменные, используемые в программном коде и до сих пор не указанные в качестве вводов или выводов, создаются автоматически в ходе сохранения или проведения синхронизации вручную. Они приводятся здесь вместе со своими дополнительными свойствами.

Name	Category	Data Type / Pin Type	Default value	Dflt. used	Dim.	Rem.	RR	Usage	Trans.
E	IEC type	REAL	0	✓		✓	R W	Ctrl	!
E_last	IEC type	REAL	0	✓		✓	R W	Ctrl	!
U_d	IEC type	REAL	0	✓		✓	R W	Ctrl	!
U_i	IEC type	REAL	0	✓		✓	R W	Ctrl	!
U_p	IEC type	REAL	0	✓		✓	R W	Ctrl	!

Рисунок 60: Вкладка 'Local variables' (Локальные переменные) функционального блока (ST)



Все вкладки можно открыть в отдельном окне, выбрав в контекстном меню пункт '**'Separate Window'** (Отдельное окно). Эта функция позволяет сохранять общее представление в ходе разработки в САЕ.



Методы работы с редактором ST уже были описаны в учебном модуле. Работа с редактором ST здесь осуществляется так же, как и в контексте проекта. См. раздел [Работа с редактором ST](#).

## 6.5 Использование блоков Automation Studio (AS) в функциональном блоке ST

Для локальных переменных можно использовать не только основные типы данных IEC (из списка основных типов данных для контроллера), но также и определения структур или типы внешних функциональных блоков.

Параметры, необходимые для этого, настраиваются в столбце 'Category' (Категория) в локальных переменных.

**Впоследствии будет приведен пример использования функционального блока AS 'RF\_TRIGGER' из библиотеки AS 'standard'.**



Обзор функций и функциональных блоков можно найти в справочной документации по Automation Studio (Programming / Debugging & Diagnosis / Libraries (Программирование / Отладка и диагностика / Библиотеки)).

На соответствующую библиотеку AS необходимо создать ссылку в свойствах библиотеки САЕ, на вкладке '**'AR OS versions'** (Версии ОС AR), чтобы можно было в коде осуществлять вызов функций и функциональных блоков из Automation Studio.

## ST в разработке библиотек

После нажатия кнопки [Dependencies (Зависимости)] в появившемся диалоговом окне 'AR-OS version dependencies' (Зависимости версии ОС AR) необходимо открыть окно выбора системных модулей (кнопка [System modules (Системные модули)]), затем выбрать необходимую библиотеку AS (в нашем примере 'standard' ) и вставить в список 'Module dependencies' (Зависимости модулей).

**После этого необходимо выполнить процедуру 'Build all (library)' (Собрать все (библиотека)).**

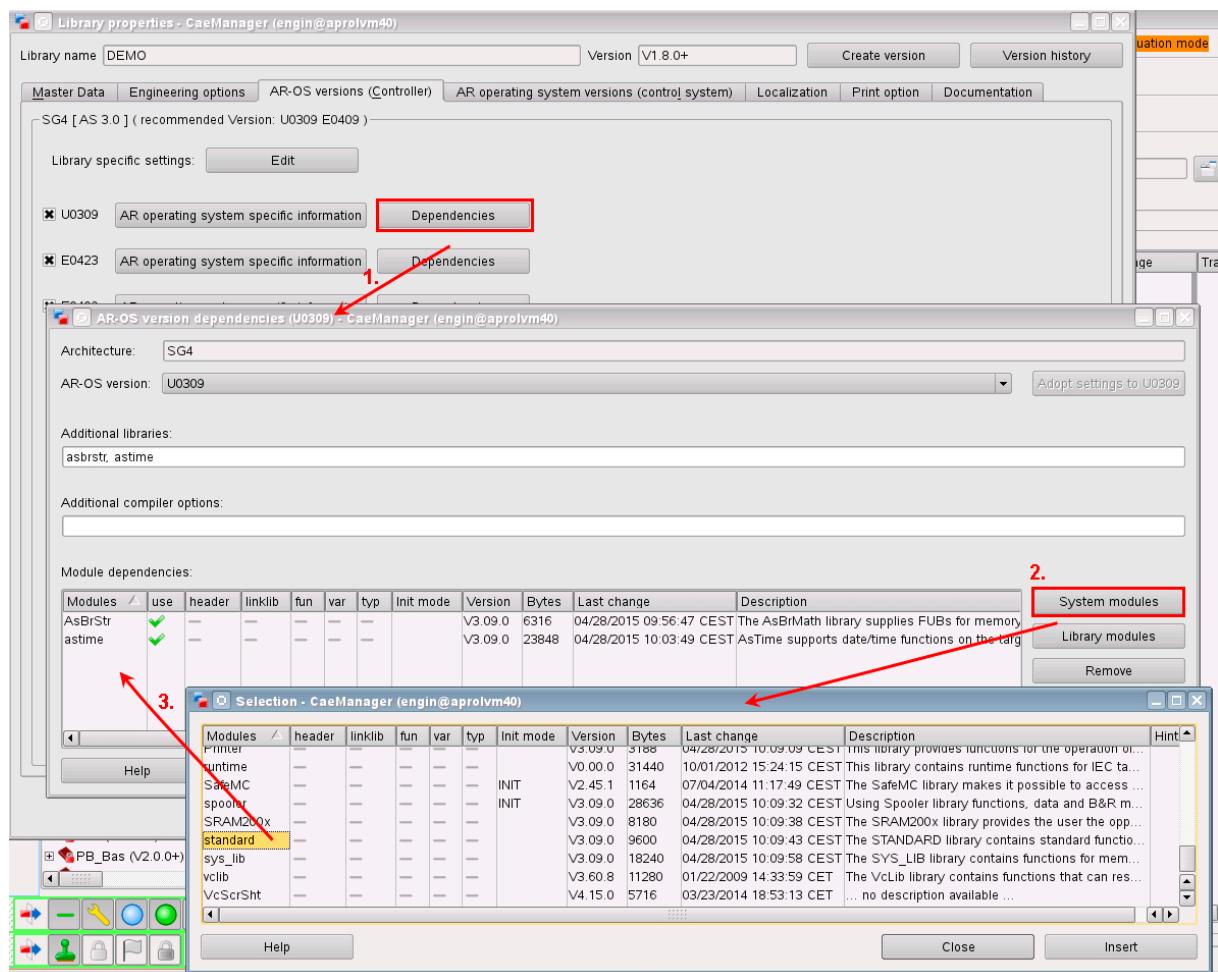


Рисунок 61: Добавление библиотеки AS 'standard' (Свойства библиотеки САЕ)

На вкладке 'Local variables' (Локальные переменные) необходимо задать следующие параметры:

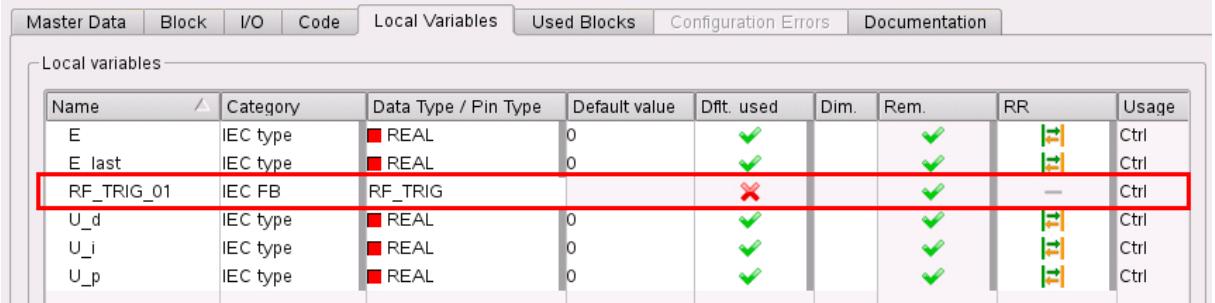
Столбец	
Name (Имя)	Имя экземпляра функционального блока, по которому в коде будет осуществляться его вызов. <i>В примере: 'RF_TRIG_01'</i>
Category (Категория)	Выберите из раскрывающегося списка запись 'IEC-FB'.
Pin type/data type (Тип контакта/тип данных)	Введите имя функционального блока из библиотеки. <i>В примере: 'RF_TRIG'</i>

**Столбец**

Description (Описание)

Здесь пользователь может ввести описание переменной.

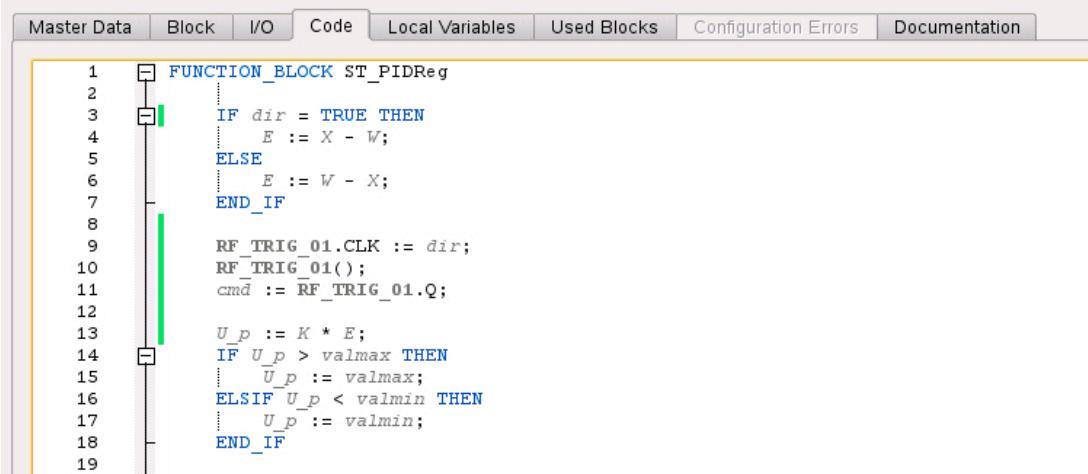
**Создавать дополнительные записи и вносить какие-либо изменения в другие столбцы для нашего примера не требуется.**



Name	Category	Data Type / Pin Type	Default value	Dfnt. used	Dim.	Rem.	RR	Usage
E	IEC type	REAL	0	✓		✓	Ctrl	
E_last	IEC type	REAL	0	✓		✓	Ctrl	
RF_TRIG_01	IEC FB	RF_TRIG		✗		✓	—	Ctrl
U_d	IEC type	REAL	0	✓		✓	Ctrl	
U_i	IEC type	REAL	0	✓		✓	Ctrl	
U_p	IEC type	REAL	0	✓		✓	Ctrl	

Рисунок 62: Создание экземпляров функциональных блоков AS на вкладке 'Local variables' (Локальные переменные)

Теперь настроенный экземпляр функционального блока AS может быть вызван в коде.



```

1  FUNCTION_BLOCK ST_PIDReg
2
3  IF dir = TRUE THEN
4      E := X - W;
5  ELSE
6      E := W - X;
7  END_IF
8
9  RF_TRIG_01.CLOCK := dir;
10 RF_TRIG_01();
11 cmd := RF_TRIG_01.Q;
12
13 U_p := K * E;
14 IF U_p > valmax THEN
15     U_p := valmax;
16 ELSIF U_p < valmin THEN
17     U_p := valmin;
18 END_IF
19

```

Рисунок 63: Вызов блока 'RF\_TRIG\_01' в коде



Также в блоках библиотеки ST можно использовать существующие блоки библиотек САЕ (см. [Вызов существующих блоков САЕ в ST](#)).

## 6.6 Практическое задание – кран

### Задача: общая нагрузка крана

Кран может поднимать одновременно пять грузов. Каждый датчик нагрузки подключен к аналоговому входу и возвращает значения в диапазоне от 0 до 32767. Для вычисления общей нагрузки и среднего значения необходимо сложить отдельные значения нагрузок, а затем разделить результат на количество датчиков нагрузки. Создайте решение данной задачи, используя цикл FOR.

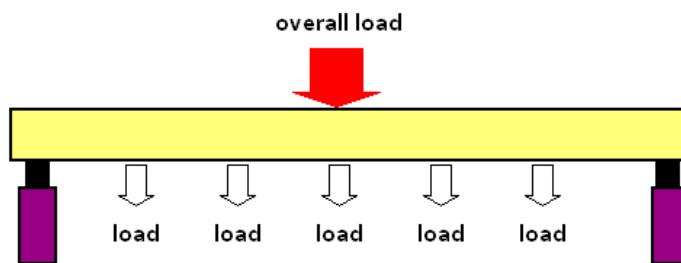


Рисунок 64: Кран с пятью грузами

#### Объявление:

```
VAR
    aWeights : ARRAY [0..4] OF INT;
    iCnt : USINT;
    sumWeight : DINT;
    avgWeight : INT;
END_VAR
```

Таблица 4: Пример объявления переменных

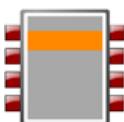


Для решения задачи создайте функциональный блок ST, содержащий массивы. Переменные массива могут быть только локальными – просто введите нужную длину в столбец "Dimensions" (Размеры).

## 7 SFC в разработке библиотек

### 7.1 Введение и функции редактора

В библиотеках APROL в целях образования структуры и для обеспечения повторного использования кода в пределах программы могут быть созданы следующие части проекта:



Функциональные блоки (SFC)

Эти функциональные блоки имеют отрисовку в виде блока, хотя это не является обязательным требованием для использования в программах ST и SFC. И благодаря этому функциональные блоки SFC также могут использоваться в CFC при помощи графического программирования.



Помимо указания основных данных и вида блока редактор SFC обеспечивает возможность настройки таких же параметров, как для программ SFC в проекте. Соответствующая информация приведена в разделе [Введение и функции редактора](#).

**Ниже будут описаны только вкладки, отличающиеся от вкладок редактора программы SFC.**

### 7.2 Создание функционального блока SFC

В контексте создания функциональных блоков SFC необходимо выполнение следующих действий:

Шаг	Разработка
1	Создание функционального блока SFC
2	Настройка основных данных
3	Создание сети SFC
4	Объявление вводов и выводов

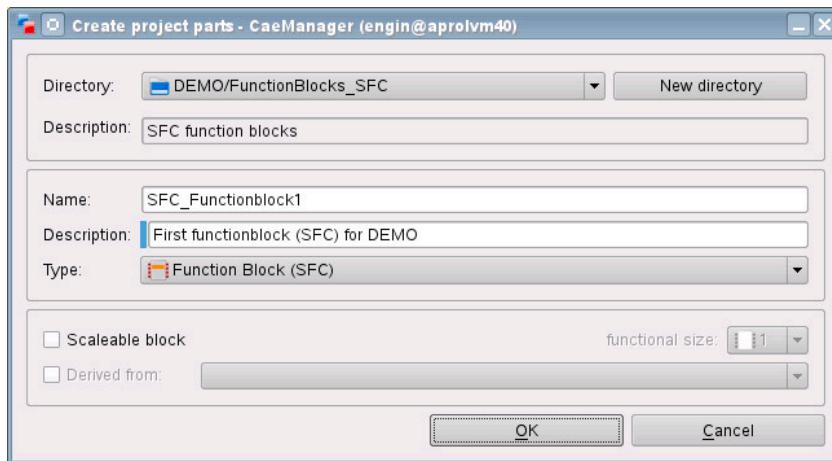


Рисунок 65: Создание функционального блока SFC

# SFC в разработке библиотек

Навигация: CaeManager / CAE library / File / New / Function Block (SFC) (CaeManager / Библиотека САЕ / Файл / Создать / Функциональный блок (SFC))



Name (Имя)

<имя блока>

Подтвердите нажатием кнопки [OK]

## 7.3 Основные данные функционального блока SFC

The screenshot shows the 'Master Data' tab of the SFC editor. The 'Name' field is set to 'SFC\_Functionblock1'. The 'Description' field contains the text 'First functionblock (SFC) for DEMO'. The 'General options' section includes checkboxes for 'Block for Controller' (checked), 'Conforming to IEC' (checked), 'Block for Control Computer' (unchecked), and 'Library internal' (unchecked). The 'Remanence / Redundancy' section shows 'Remanence' set to 'Off-state' and 'Controller redundancy capability' set to 'Not Evaluated'. The 'Announcement of discontinuation' section has checkboxes for 'Discontinued', 'Error message for usage', and 'Replacement'. The 'Library' dropdown is set to 'DEMO' and the 'Project part' dropdown is set to 'SFC\_Functionblock1'. The 'Scaleable block' section has checkboxes for 'Scaleable block' and 'Derived from'. The 'SFC options' section includes a checkbox for 'Controllable by SFCViewer' (checked), a radio button for 'Name' (selected), and checkboxes for 'Manual initialization of SFC' and 'Short description'. There are also settings for 'Step / Transition labeling (default)', 'Object size', 'Grid settings', 'Width', 'Height', 'Horizontal spacing', and 'Vertical spacing'.

Рисунок 66: Вкладка "Основные данные" (Master Data)

На первой вкладке редактора SFC “Master Data” (Основные данные) доступны следующие параметры, действующие в пределах библиотеки.

- Объект SFC всегда требует наличия **контроллера** и всегда **соответствует IEC**, поэтому соответствующие параметры всегда включены. Пользователь может только отметить эту часть проекта как внутреннюю для библиотеки.
- Remanence (энергонезависимость)**: здесь можно отметить весь блок, т. е. все его выводы, в качестве реманентного.
- Чтобы блок мог использоваться на резервируемом контроллере, необходимо включить параметр **“Controller Redundancy Capability”** (**Совместимость с резервируемым контроллером**).
- Ниже блок можно **отменить**, используя при этом сообщение об ошибке и указанием на замещающий блок; ещё ниже настаивается возможность **масштабирования**.
- Подготовка SFC к последующему влиянию на его выполнение при помощи SFCViewer осуществляется включением параметра **'Controllable by SFCViewer'** (**Управляется SFCViewer**). Помимо этого здесь можно настроить запуск **инициализации вручную**.
- Также здесь можно установить параметры **размера** элементов SFC и **компоновки** сети SFC.

## 7.3.1 Инициализация функционального блока SFC вручную

Инициализация вручную означает, что в процессе выполнения функциональный блок SFC запускается не автоматически, а при помощи определенного триггера.

Параметр '**Manual initialization**' (**Инициализация вручную**) функционального блока может быть включен/отключен на вкладке 'Master data' (Основные данные) при помощи соответствующего флашка.

Таким образом, функциональный блок запустится только после того, как значение переменной 'SFCInit' / 'SFCReset' в программе CFC, SFC или ST изменится с 'TRUE' на 'FALSE' и обратно на 'TRUE' (передний и задний фронт сигнала).

При включении параметра 'Manual initialization' линия ввода (контакт) 'SFCInit' / 'SFCReset' помечается как требующая соединения в обязательном порядке.

При включенном параметре 'Manual initialization' имена контактов отображаются полужирным шрифтом.

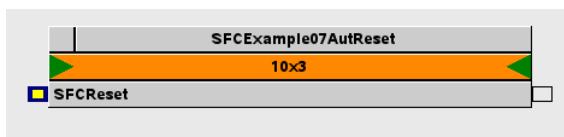


Рисунок 67: Линия ввода (контакт) 'SFCReset'

При автоматической инициализации блока имена контактов 'SFCInit' или 'SFCReset' выделяются на представлении блока зеленым цветом, как и на представлениях при использовании в гипермакросе или CFC.

Использование параметра 'Controllable by SFCViewer' (Управляется SFCViewer) отмечается на компоновке блока двумя зелеными треугольниками.

Активация параметра позволяет выбирать средства управления в SFCViewer и редакторе SFC.



При создании функционального блока для него по умолчанию настроена автоматическая инициализация.

## 7.4 Графическая настройка функционального блока

Графическая настройка сети SFC производится на вкладке 'Chart' (Схема).



Информация о графической настройке, а также создании и использовании действий приведена далее в тексте модуля, в разделе [Графическая настройка сети SFC](#).

# SFC в разработке библиотек

## 7.5 Использование контактов и переменных

Существует множество способов использования переменных в функциональном блоке (SFC):

- В качестве вводов и выводов в представлении блока
- В качестве вводов и выводов в представлении списка вводов/выводов
- В качестве локальных переменных

### Вводы и выводы в представлении блока

Линии ввода и вывода (контакты) указываются так же, как всегда в APROL, на вкладке 'Block' (Блок), и таким образом создается компоновка блока.

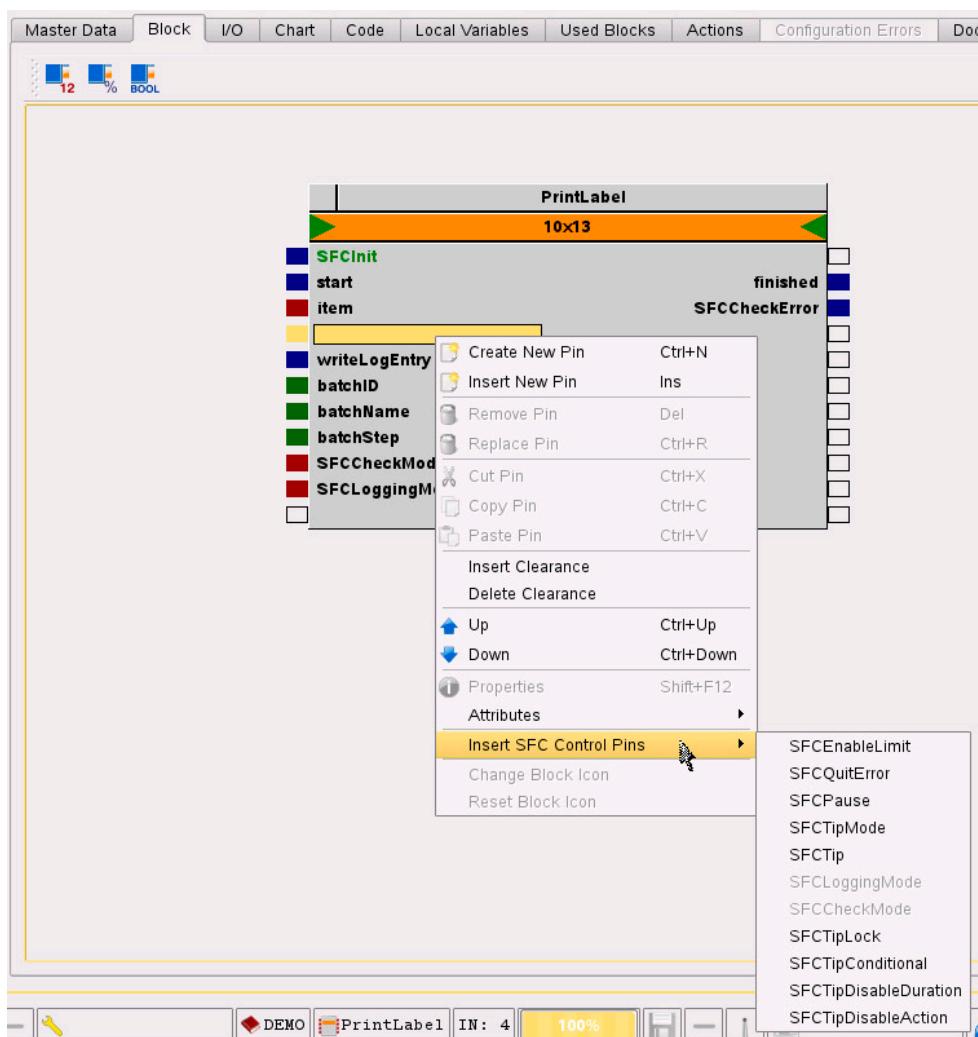


Рисунок 68: Вкладка "Блок" функционального блока

Для создания нового контакта выберите пункт контекстного меню "**Create New Pin (Создать новый пин)**" в свободной части блока. Если свободное место отсутствует, перетащите вниз нижнюю границу блока.

Для контекстных данных схем SFC используются **типы пинов "SFCContext"**. Выбор осуществляется при помощи пункта контекстного меню '**Create New Pin**' (Создать контакт) / радиокнопкой '**Pin type**' (Тип контакта) / [...]

Назначение управляющих переменных выполняется при помощи пункта контекстного меню '**Insert Control Variables**' (Вставить управляющие переменные).

### Вводы и выводы в представлении списка вводов/выводов

Все входные и выходные контакты блока перечислены на вкладке '**I/O (Ввод/вывод)**'. Новые контакты можно создать при помощи пункта контекстного меню 'Insert new pin' (Вставить новый контакт).

The screenshot shows the 'I/O' tab of a functional block configuration. It contains two tables:

- Input pins:**

Pos.	Name / Control value	IEC Type / Pin Type	Default Value	Unit	Description
1	SFCInit	BOOL	False (0)		Init SFC
2	start	BOOL	False (0)	049:	
3	item	INT	0	049:	
5	writeLogEntry	rc_confirm (BOOL)	False (0)	049:	
6	batchID	rc_batchid (SSTRING)	0	049:	
7	batchName	rc_batchname (SSTRING)	0	049:	
8	batchStep	rc_batchstep (SSTRING)	0	049:	
9	SFCCheckMode	SFCCheckMode (USINT)	Only set error flag (1)		Sets the mode for the se
10	SFCLoggingMode	SFCLoggingMode (USINT)	Deactivate logging (0)		Enables logging for the S
- Output pins:**

Pos.	Name / Control value	IEC Type / Pin Type	Unit	Description	Rem.	RR	Dyn.	Sys.	V
2	finished	BOOL	049:		✗	✗	✓	✗	
3	SFCCheckError	BOOL			✗	✗	✓	✓	

Рисунок 69: Представление входов/выходов функционального блока (SFC)

Функции на этой вкладке аналогичны функциям предыдущей вкладки '**Block (Блок)**'.

### Локальные переменные

Переменные, используемые в программном коде и до сих пор не указанные в качестве вводов или выводов, создаются автоматически в ходе сохранения или проведения синхронизации вручную. Они приведены здесь вместе с их свойствами.

The screenshot shows the 'Local variables' tab of a functional block configuration. It contains a table:

Name	Category	Data Type / Pin Type	Default value	Dflt. used	Dim.	Rem.	RR	Usage	Trans.	Des.
LocOutput	IEC type	DWORD	0x01	✓		✗	✗	Ctrl	✗	
SaveVal	IEC type	LREAL	0	✓		✗	✗	Ctrl	✗	
Toggle	IEC type	BOOL	False (0)	✓		✗	✗	Ctrl	✗	049

Рисунок 70: Вкладка 'Local variables' (Локальные переменные) функционального блока



Все вкладки можно открыть в отдельном окне, выбрав в контекстном меню пункт '**Separate Window**' (Отдельное окно). Эта функция позволяет сохранять общее представление в ходе разработки в САЕ.

## 7.6 Вызов функций и функциональных блоков

В контексте библиотеки можно создать локальные функции и функциональные блоки, использовать блоки библиотек САЕ и функциональные блоки Automation Studio.



Функции редакторов ST и SFC в данном отношении совпадают.

Для того, чтобы узнать, как создавать и использовать **локальные функции и функциональные блоки**, обратитесь к соответствующей главе в разделе данного модуля, посвященному структурированному тексту: [Локальные функции и функциональные блоки](#)

Из следующих разделов, посвященных структурированному тексту, вы узнаете, как использовать **существующие блоки библиотек** (см. главу [Вызов существующих блоков САЕ в ST](#)) в функциональном блоке (SFC) и **функциональные блоки Automation Studio** (см. главу [Использование блоков Automation Studio \(AS\)](#) в функциональном блоке ST).

## 7.7 Практическое задание – управление емкостью

Необходимо организовать управление уровнем и температурой воды в аквариуме. Управление включает в себя два процесса, которые должны рассматриваться как отдельные.

Нагревающий элемент должен повышать температуру воды до тех пор, пока она не достигнет заданного значения. По достижении заданного значения нагревательный элемент должен быть выключен.

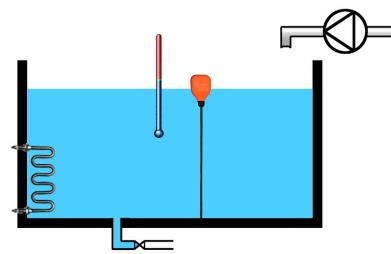


Рисунок 71: Схематический вид упражнения

Управление уровнем заполнения осуществляется при помощи насоса и поплавкового реле. Как только поплавковое реле обнаруживает низкий уровень воды, насос должен начать подавать в аквариум воду и продолжать до тех пор, пока аквариум не будет заполнен. В целях сокращения количества включений/отключений насоса в связи с образованием волн на поверхности воды добавляется такое значение, как время до повторного заполнения.

### Применение SFC

Поскольку задача включает в себя два различных процесса, ее решение может быть реализовано при помощи двух параллельных ветвей, выполняющихся одновременно. Первая из этих ветвей отвечает за управление температурой, вторая – за регулировку уровня воды

### Реализация управления уровнем воды

Описанную задачу необходимо решить при помощи одного функционального блока (SFC) с одной параллельной ветвью. Каждая ветвь содержит состояния, которые могут быть описаны на схемах состояний.

- 1) Создайте функциональный блок SFC
- 2) Создайте параллельную ветвь.
- 3) Создайте шаги и прыжки.
- 4) Объявите переходы.

Переходы могут использоваться для оценки текущей температуры относительно заданной.  
Типом данных результата должен являться тип BOOL

- 5) Выполните программирование действий и команд переключения.

Команды для включения и отключения нагревательного элемента и насоса должны быть реализованы во входных и выходных действиях шагов.

- 6) Для временной задержки используйте действие IEC.

Переход, который деактивирует шаг ожидания, должен переключаться при помощи действия с определителем "D". Время задержки должно составлять 5 секунд.

- 7) Проверьте последовательность в автоматическом режиме, режиме подсказок или режиме форсирования.

Одно из возможных решений приведено в приложении

# Заключение

## 8 Заключение

Чего мы достигли на данный момент?

Теперь мы обладаем знаниями о языках программирования 'Structured Text' (структурированного текста) и 'Sequential Function Chart' (последовательных функциональных схем). Оба языка являются нормальными языками программирования высокого уровня, используются на программируемых логических контроллерах и имеют широкий набор функциональных возможностей.

**Structured Text (язык структурированного текста)** содержит все необходимые для использования инструменты. Этот язык программирования особенно полезен при использовании арифметических функций и создании формул для математических вычислений.

**Sequential Function Chart (язык последовательных функциональных схем)** – это понятный и хорошо структурированный язык, обладающий графическим интерфейсом пользователя. SFC отлично подходит для реализации схем состояния и конечных автоматов в форме шагов и переходов. Программы SFC можно легко диагностировать и поддерживать в актуальном и работоспособном состоянии, используя разнообразные возможности диагностики.

Область функций в ST и SFC дополняется повторяющимися действиями и функциональными блоками.

Возможность применения в сочетании с другими языками программирования IEC (например, на языках Continuous Function Chart или Structured Text) делает SFC мощным языком программирования.



Рисунок 72: Дополнительный инструмент диагностики: SFCViewer

## 9 Приложение

### 9.1 Пример программ ST для решения задач из текста модуля

#### Задача: проверка уровня жидкости, часть 1

Local Variables				
Name	Used	Category	Pin Type/Data Type	
Alarm	✓	IEC type	BOOL	
DoMeasure	✓	IEC type	BOOL	
LevelAnalog	✓	IEC type	INT	
LevelHigh	✓	IEC type	BOOL	
LevelLow	✓	IEC type	BOOL	
LevelOk	✓	IEC type	BOOL	
LevelPercent	✓	IEC type	USINT	

Объявление:	<pre>(*scaling the analog input to percent*) LevelPercent := INT_TO_USINT((LevelAnalog / 32700)*100);</pre>
Программный код:	<pre>IF DoMeasure THEN      CASE LevelPercent OF         0:             Alarm := TRUE;         1..24:             LevelLow := TRUE;         25..90:             LevelOk := TRUE;         ELSE             LevelHigh := TRUE;     END_CASE ELSE     Alarm := 0;     LevelLow := 0;     LevelOk := 0;     LevelHigh := 0; END_IF</pre>

Таблица 5: Оператор CASE для поочередной оценки значений и диапазонов значений

## Приложение

### Задача: проверка уровня жидкости, часть 2

Instance/Variable	Used	Remanent	RR	Pin Type/FB/I	I/O Constant	I/O
AlarmDelayTON	✓			IEC61131_3_TON		
AlarmDelayTON_r.ET		remanent		TIME	-	
AlarmDelayTON_r.Q		remanent		BOOL	-	
AlarmDelayTON_v.IN		volatile		BOOL	- False (0)	
AlarmDelayTON_v.PT		volatile		TIME	- T#5s	

**Используемые блоки:**

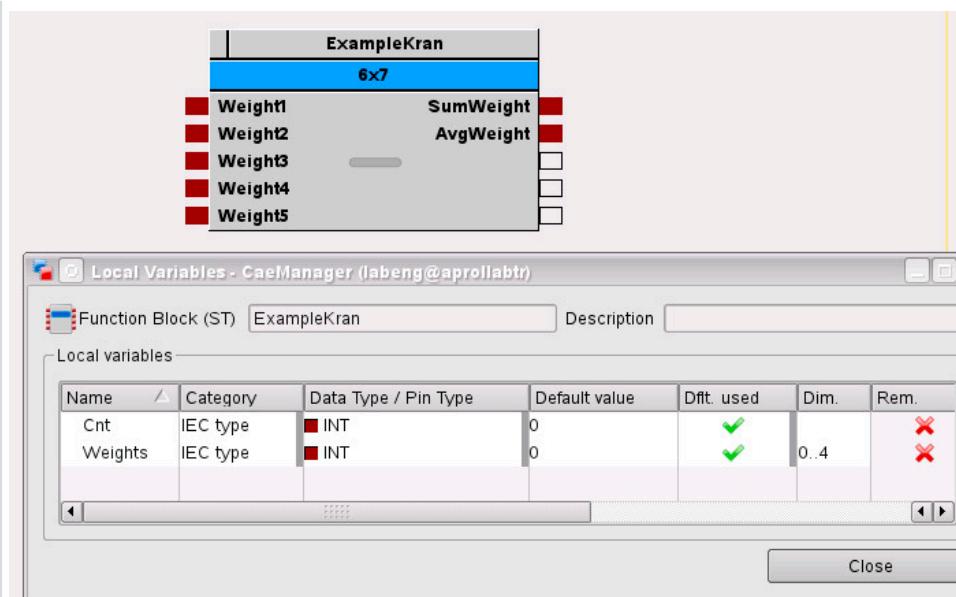
```
(*scaling the analog input to percent*)
LevelPercent := INT_TO_USINT((LevelAnalog / 32700)*100);
IF DoMeasure THEN
    CASE LevelPercent OF
        0:
            AlarmDelayTON_v.IN := TRUE;
            IEC61131_3_TON( AlarmDelayTON_v, AlarmDelayTON_r );
            Alarm := AlarmDelayTON_r.Q;
        1..24:
            LevelLow := TRUE;
        25..90:
            LevelOk := TRUE;
        ELSE
            LevelHigh := TRUE;
    END_CASE
ELSE
    Alarm := 0;
    AlarmDelayTON_v.IN := 0;
    IEC61131_3_TON( AlarmDelayTON_v, AlarmDelayTON_r );
    LevelLow := 0;
    LevelOk := 0;
    LevelHigh := 0;
END_IF
```

**Программный код:**

Таблица 6: Расширение части 1 с использованием блока TON

**Задача: общая нагрузка крана**

**Объявление:**



**Программный код:**

```

FUNCTION_BLOCK ExampleKran
    Weights[0] := Weight1;
    Weights[1] := Weight2;
    Weights[2] := Weight3;
    Weights[3] := Weight4;
    Weights[4] := Weight5;

    FOR Cnt := 0 TO 4 DO
        SumWeight := SumWeight + Weights[Cnt];
    END_FOR

    AvgWeight := DINT_TO_INT(SumWeight / 5);

END_FUNCTION_BLOCK

```

Таблица 7: Оператор FOR, сложение масс грузов

## Приложение

### 9.2 Пример программ SFC для решения задач из текста модуля

#### Пример решения задачи со смесителем

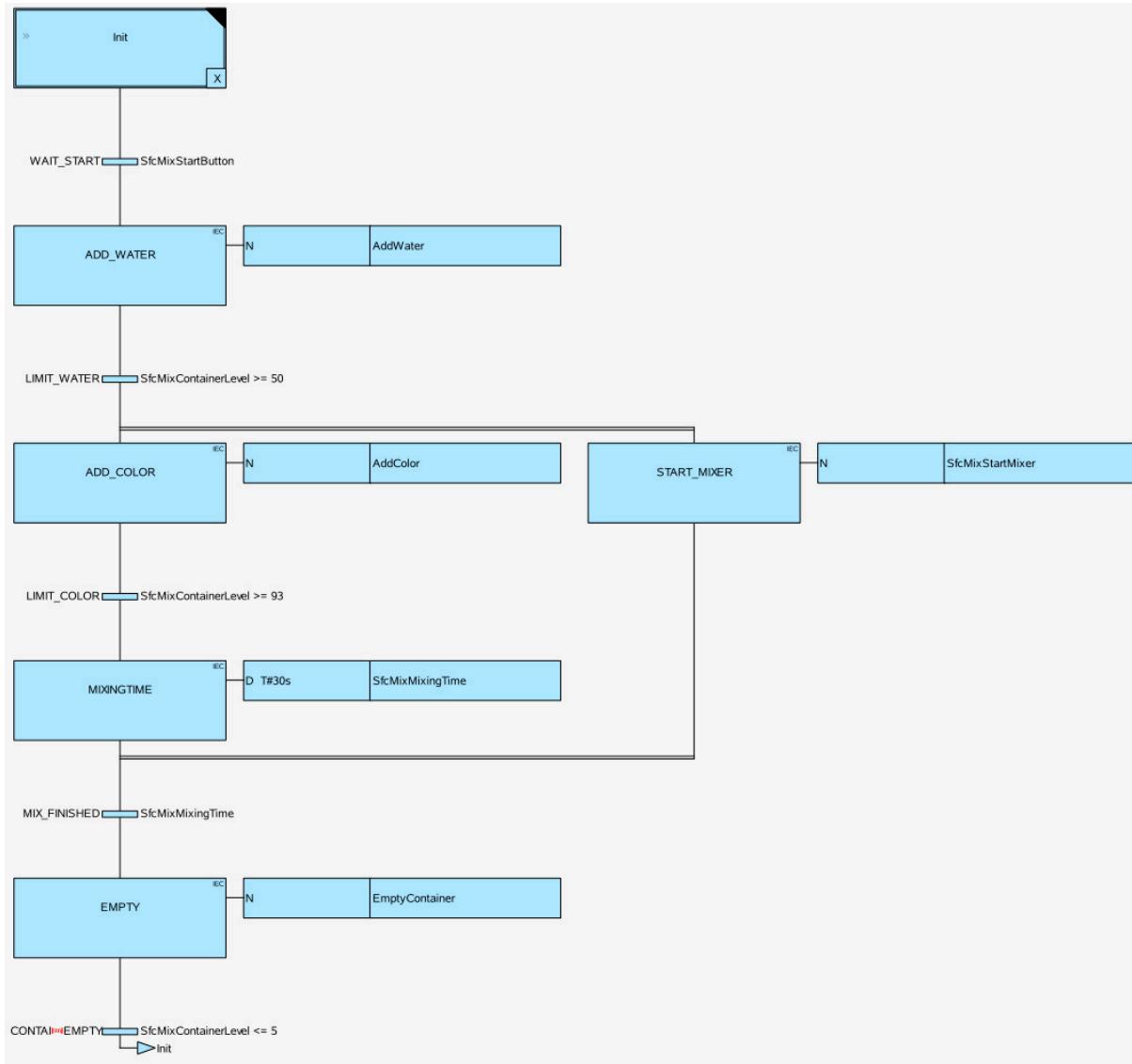


Рисунок 73: Схема SFC для смесителя

### Решение задачи: управление уровнем воды

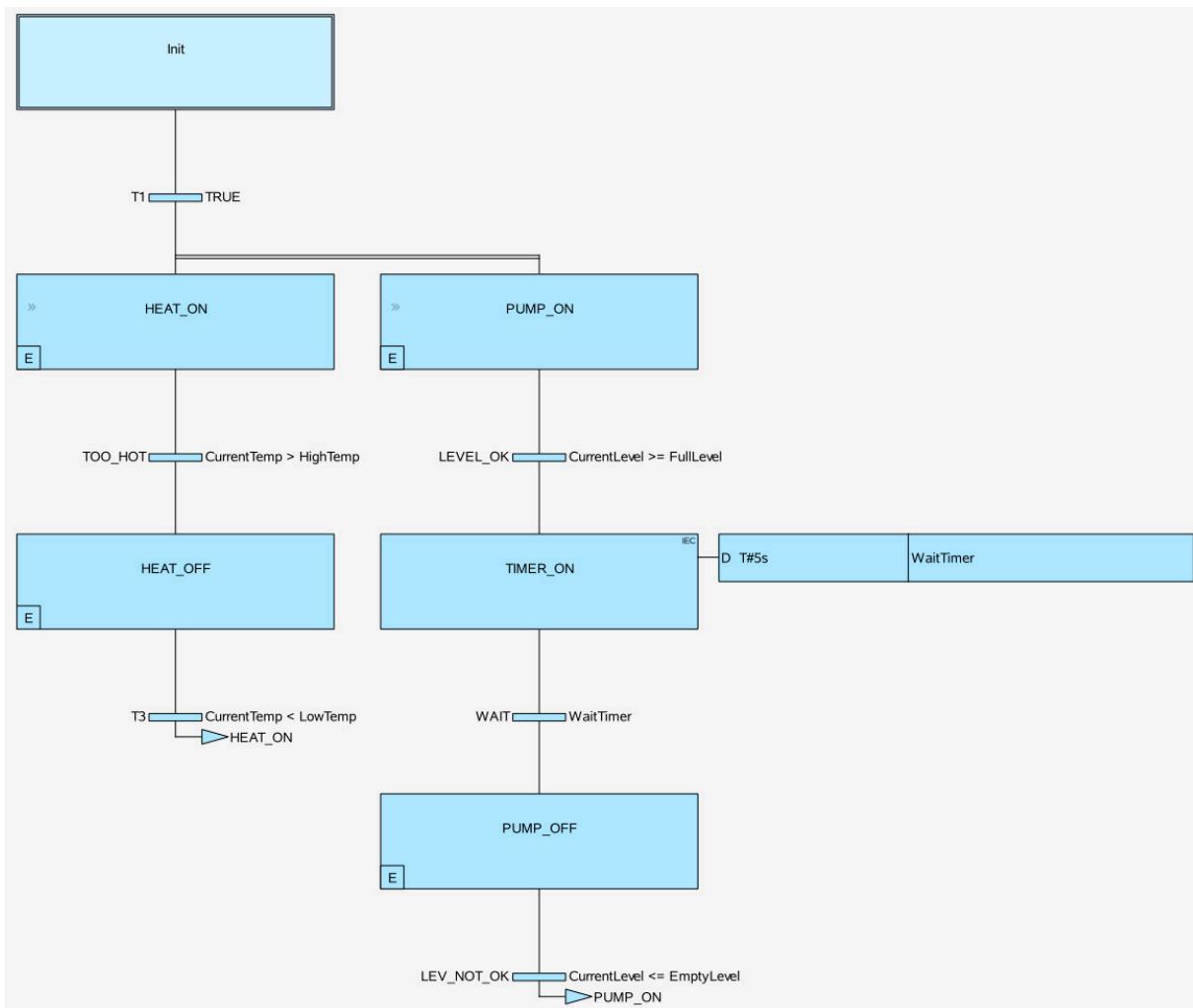


Рисунок 74: Схема SFC для управления аквариумом

# Представлено Академией Автоматизации

## Представлено Академией Автоматизации

Академия автоматизации (Automation Academy) предоставляет целевые учебные курсы для наших клиентов, а также наших собственных сотрудников.

### B Automation Academy Вы повысите свои навыки в кратчайшие сроки!

Наши семинары помогут Вам углубить Ваши знания в области промышленной автоматизации.

После их завершения Вы сможете реализовать эффективные решения автоматизации с использованием технологии B&R. Гарантируйте себе конкурентное преимущество, чтобы Вы всегда могли оперативно реагировать на постоянно изменяющиеся тенденции рынка.



### Семинары



Качество и актуальность – это важные компоненты наших семинаров. Темпы обучения основаны на опыте участников курсов, и учитывают требования, с которыми они сталкиваются. Комбинация работы в группе и самообучения обеспечивает высокий уровень гибкости, необходимый для максимально эффективного обучения. Каждый семинар проводится одним из наших высоко квалифицированных и опытных преподавателей.

### Учебные модули

Наши учебные модули являются основой для обучения на семинарах, а также для самостоятельного изучения. Эти компактные модули основаны на согласованной дидактической структуре. Их структурированное представление "снизу вверх" позволяет быстро и эффективно изучать сложные, взаимосвязанные темы. Они представляют отличный дополняющий материал к нашей обширной справочной системе. Учебные модули доступны как для скачивания с сайта, так и могут быть заказаны в печатном виде.

### Разделение по предметной области:

- ⇒ Технология управления
- ⇒ Управление движением
- ⇒ Технология безопасности
- ⇒ Визуализация и управление
- ⇒ Управление тех.процессами
- ⇒ Диагностика и обслуживание
- ⇒ POWERLINK и openSAFETY

### Система ETA



Система ETA представляет собой программно-технологический комплекс для обучения, практических занятий и лабораторных испытаний. Система представлена конструкциями двух типов. Система ETA light весьма компактна и легко транспортируется, не занимает много места и идеально подходит для лабораторных работ. Система ETA standard имеет прочную механическую конструкцию и включает в себя ряд предустановленных датчиков и исполнительных механизмов.

### Узнайте больше!

Остались какие-либо открытые вопросы для дальнейшего изучения? Заинтересовались тем, что предлагает Академия Автоматизации B&R? Вы обратились прямо по адресу.

Следующие ссылки предоставляют дополнительную информацию:  
[www.br-automation.com/academy](http://www.br-automation.com/academy)





V1.4 ©2019/10/15 B&R, Все права защищены.  
Все представленные торговые марки являются собственностью соответствующих компаний.  
Мы оставляем за собой право вносить технические изменения без уведомления.

